

ADA020515

DISTRIBUTION STATEMENT
Approved for public release;
Distribution Unlimited

DDC
RECEIVED
FEB 17 1976
REGISTRATION

9 TECHNICAL REPORT NUMBER 106
THEMIS REPORT NO. 31

14 TR-106,
THEMIS-UGA-31
format

6 AN INTERACTIVE WORKSHEET SYSTEM
FOR STATISTICAL USAGE.

5

BY

10 Stephen F./Bingham
and
Rolf E./Bargmann

12 298p.

Reproduction in whole or in part is permitted for
any purpose of the United States Government. This
Research was supported, in part, by the Office of
Naval Research, Contract No. N00014-69-A-0423

16 NR042-261

15

Rolf Bargmann
Principal Investigator

The University of Georgia
Department of Statistics and Computer Science

Athens, Georgia

11 Aug 1975

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DDC
RECEIVED
FEB 17 1976
RECEIVED
A

405 576

TABLE OF CONTENTS

CHAPTER	PAGE
I INTRODUCTION	1
II INTRODUCTION TO OMNITAB	6
III IMPLEMENTATION	17
A. Problems	17
B. Documentation	26
IV THE ADDITION OF COMMANDS	127
V EXAMPLES	140
A. Order Statistics	140
B. Bioassay	146
C. Scaling of Multi-dimensional Categorized Variables	154
D. Multivariate Analysis	164
BIBLIOGRAPHY	179
APPENDIX: Program Listings	185

ACCESSION FOR	
DDC	White Section <input checked="" type="checkbox"/>
DDC	Bell Section <input type="checkbox"/>
UNCLASSIFIED	<input type="checkbox"/>
<i>Noted on file</i>	
BY	
DISTRIBUTION & ACQUISITION	
DATE	APRIL 2 1964
A	

LIST OF TABLES

TABLE	PAGE
IV-1 Conversion Codes	134

LIST OF FIGURES

FIGURE	PAGE
II-1 Library of Commands	7
II-2 Command Illustrations (partial display)	13
II-3 Illustration of several statistical commands (partial page)	16
III-1 Programmed Function Keyboard	18
III-2 Overlay Structure	19
III-3 Entering a command after seeing the initial display . . .	22
III-4 The screen when a command can be entered	23
III-5 Entering data using a READ command	25
III-6 Listing of commands entered and recovered using PROGRAM .	28
III-7 Basic Instructions	29
III-8 Section of Worksheet	30
IV-1 Load module JCL sequence	130
V-1 Worksheet after reading in the data	142
V-2 Worksheet after entering GENERATE 1. 1. *NRMAX*3	143
V-3 Worksheet after entering SUM 4 5	145
V-4 First page of instructions used in bioassay example . . .	147
V-5 Worksheet after entering YORMP 4 5	149
V-6 Worksheet after entering DIVIDE 27 26 21 B	150
V-7 Worksheet after entering SUBTRACT 14 16 16 A	151

LIST OF FIGURES (cont.)

FIGURE	PAGE
V-8 Worksheet after entering YORMX 26 7 PREDICTED PROPORTION	152
V-9 Worksheet after entering DIVIDE *10,8* 15 14 YBAR	153
V-10 First page of commands for scaling example	156
V-11 Second page of commands for scaling example	157
V-12 Worksheet after entering MADD 29,1,1,2,31,1,34,1	158
V-13 Worksheet after entering in columns 9 and 10 the values of .061093 and .9981321	159
V-14 Second page of commands for scaling example	161
V-15 Worksheet just before entering MMULT 1,9,1,2,1,3,2,2,5,9 .	162
V-16 Third page of commands for scaling example	163
V-17 Worksheet after entering correlation matrix	167
V-18 Worksheet after entering correlation matrix	168
V-19 Changing the row counter while entering data	169
V-20 Worksheet after entering MMULT 10,1 2,2 17,1 2,2 20,1 . .	171
V-21 Worksheet after entering MMULT 10,1 2,2 17,1 2,2 20,1 THIS IS PSTAR	175
V-22 Commands used for correlation example	176
V-23 A second approach to the correlation example	177

CHAPTER I

INTRODUCTION

With the proliferation of packages and systems for computer based statistical analysis it appears redundant to develop new packages or expand existing ones. Where methods of analysis are well standardized, as in most applications of the general linear model, or in experimental design, such work would seem superfluous.

There are, however, many methods, either complex or new, for which efficient algorithms are not easily available and which, even for the details of data analysis, require mathematical insight. Careless analysis, in such instances, may produce ridiculous results. Even the experienced statistical analyst needs to experiment with different algorithms in such cases. As Norvin Muller has said, "Much of data analysis involves iterative processing using non-quantitative insight and an artistic flair for looking at data." [46] The availability of an integrated mathematical and statistical computer system greatly facilitates such data analysis. Examples of such applications are correlational analysis, structural or factor analysis, Bayesian inference, determination of confidence and tolerance regions, estimation problems in stochastic processes, non-linear estimation, response surface estimation, calibration of instruments, and many others. However, the traditional packages, invariably based on design, least

squares, normal equations (and, very rarely, some graphical displays), are inadequate for such problems.

Lacking an integrated system, one might consider using the algorithmic packages and languages presently available for mathematical analysis. However, even the most developed and best documented units (e.g. APL/360[25]) provide only a quite primitive level of algorithms, not even including that basic set of routines (e.g. those presented by Carnahan, Luther and Wilkes [14]) which form the common stock of introductory courses in scientific computation. For example, many polynomial regression subroutines fail to use orthogonal polynomials and subroutines for solving linear equations by Gaussian elimination fail to accumulate double precision inner products. [36] In fact, among the hundreds of available computer programs and systems it is not easy to find one which is particularly well adapted for the application of exploratory techniques. Although it is obvious that array operations are crucial, the matrix-interpretive languages so often used in computer systems are not the ideal solution. An interactive system based on a worksheet, of which portions can be inspected at every single operation, seems to be more promising; of course, matrix operations should be included in such array-manipulative programs. OMNITAB [31,37] is a programming system that is based on the use of such a data worksheet. Its set of commands can be divided into two basic groups. One group of commands can be used to process columns of data, while the other group enables one to handle analyses involving matrix manipulation.

Development of OMNITAB was initiated by Joseph Hilsenrath in the early 60's at the National Bureau of Standards. He saw OMNITAB as a

computational tool directed at the user who was accustomed to performing most calculations on a desk calculator. With the advent of very inexpensive pocket calculators, the number of users of this type has become larger. It enables such a person to perform calculations on a computer very quickly and easily without requiring him to also learn a complex programming language.

The original version, which was written for the IBM 7094, is described in detail by Hilsenrath et.al.[31] However, this version was written primarily in assembly language and thus was machine dependent. Later it was rewritten in FORTRAN and implemented on a UNIVAC 1108. This FORTRAN version has since been implemented on an IBM 360, Burroughs 5500 and CDC 6600 [37,38]. The IBM 360 implementation was carried out by R. L. Chamberlain who also wrote a very helpful user's guide [37] and an operating systems manual [15].

Since its introduction, OMNITAB has been used extensively in a variety of fields by many different users. Applications can be found in molecular spectroscopy [4,39,40,44,45], agronomy [13] and photochemistry [9-11] as well as in numerous other fields. Most applications, however, involve the use of one of the two commands, FIT and POLYFIT. These two commands are used to obtain least squares fits for linear models. Numerous applications of these two commands can be found in references 18, 22, 23, 28, 30, 35, 48, and 59.

There have been several adaptations of OMNITAB to meet special purposes. PRECISE [5], developed in the late 60's, is a multiple precision version which has been used in the preparation of tables. MINITAB [50,51,52] was developed in the Statistics Department at

Pennsylvania State University. It is designed to help the student solve many of the elementary statistical problems by supplying him both a worksheet for storage of data and a special set of commands for performing the calculations. An interactive version of OMNITAB has been developed at the University of Texas at Austin for use by psychology students [58]. These last two adaptations were both made in the early 70's.

The simplicity of OMNITAB commands as well as the nature of the worksheet suggested that an interactive version of OMNITAB designed especially for use by statisticians would be helpful. The requirements for such a version included:

- (1) Immediate access to the worksheet after performance of every statement
- (2) Availability of statistical distribution functions.
- (3) A facility to replace existing commands in the system
- (4) A facility which enables a statistician who knows only FORTRAN IV programming to add new commands and to incorporate them into the system
- (5) To permit execution in a limited region of core (overlays).
- (6) To permit the instantaneous editing of data, and the correction of commands when the user makes a mistake.
- (7) To display all of the commands used during an entire work session

The purpose of this research has been to implement such an interactive version of OMNITAB and to present examples for the utilization of this interactive unit in the solution of statistical problems. After the

system became operational under the Graphics Monitor System [47] on the IBM 2250 graphics console at the University of Georgia, its general effectiveness and ease of operation was tested by having it used by students in classes on Multivariate Methods and Scientific Computation (STAT 825, Winter, 1973 & 1974; STAT 803 and 804, Fall 1973), by research workers in radioecology for the purpose of deciding on appropriate kinetic models, and by several others, especially for matrix-manipulative purposes.

Chapter II consists of a brief user's guide to OMNITAB. Chapter III presents details concerning the implementation of an interactive version of OMNITAB in the computing environment available at the University of Georgia. Many implementation problems were of course quite general. Chapter IV describes several methods which can be used for increasing the number of available commands and includes examples. Chapter V presents a number of applications of interactive OMNITAB in solving various statistical problems. These examples illustrate how interactive OMNITAB can be used in solving problems of various types and complexities by both teacher and student and as a computational tool by researchers.

CHAPTER II

INTRODUCTION TO OMNITAB

In this chapter certain basic details will be given regarding the use of the OMNITAB language. Details not covered here can be found in The OMNITAB Programming System: A Guide for Users by Jowett and Chamberlain [37].

An OMNITAB command consists basically of two parts. The first part is a key word which begins each command. These key words are listed in Figure II-1, and they constitute the basic operation codes. Each key word is limited to six letters. The second part of the OMNITAB command contains the argument list and any comments which the user inserts. It begins either in position seven or after a blank which follows a key word of less than six letters. Comments are ignored when a command is processed. For example, consider the typed message:

ADD COLUMN 1 TO COLUMN 2 AND STORE THE RESULTS IN COLUMN 3

ADD is the keyword. The argument list consists of the integers 1, 2 and 3. Since the remainder of the message is comment, the typed message:

ADD 1 2 3

would produce identical results.

For each argument in the argument list, two pieces of information are retained. First, an indicator is set to distinguish non-integer

OUTPUT AREA					
COMMANDS CURRENTLY IMPLEMENTED IN OMNITAB					
ARAD	ARCSIN	CHIP	EXPAND	N(X'AX)	NSCALAR
ARRVEC	ARCTAN	CHIX	EXPONENT	N(X'AX')	NSUB
ABS	ASCALAR	CHIZ	FFP	N(X'X)	NTRANS
ABSOLUTE	ASIN	CLUSE	FFX	N(X'X')	MULT
ACOS	ASIND	COS	FFZ	NRAD	MULTIPLY
ACOSO	ASINH	COSO	FLIP	MAX	NVECDIAG
ACOSH	ASUB	COSH	GAMP	MAXIMUM	NVECHAT
ACOT	ATAN	COT	GAMX	NDEFINE	NZERO
ACOTD	ATAND	COTD	GAMZ	NDIAG	NEGEXP
ACOTH	ATANH	COTH	GENERATE	NERASE	ORDER
ADD	NTRANS	COUNT	HIERARCH	NIDENT	PANPROD
ADDFINE	AVECARR	DEFINE	INVERT	NIN	PASUM
ADIAO	AVECDIAG	DEHOTE	LINEAR	MINIMUM	PRODUCT
ADIVIDE	AVERAGE	DEVHOR	LOG	MINVERT	PROMOTE
ERASE	AZERO	DIV	LOGE	MLINEAR	RAISE
AMOVE	BETAP	DIVIDE	LOOTEN	MHATVEC	READ
AMULT	BETAZ	DUPLICAT	N(AD)	MMOVE	RESET
ANTILOG	BETAZ	ERASE	N(AV)	MMULT	RMS
ARADISE	BLOCKTRA	EXCHANGE	N(OA)	MOVE	ROW
ARCCOS	CHANGE	EXP	N(V'A)	MRAISE	ROWSUM
ARCCOT					
READY					

REPLY AREA

Figure II-1
Library of Commands

real arguments from integer arguments. Non-integer real arguments are generally used as constants and integer arguments are generally used as column or row numbers. In addition, the value of each argument is retained.

There are several general rules which determine the type of argument(s) required for a particular command. Integer arguments are generally used as column or row numbers or as dimensions of a matrix. Real arguments, i.e. constants containing a decimal point, are generally used as constants. To illustrate this consider the following two commands:

(1) ADD 1 2 3 and

(2) ADD 1. 2 3

The first command will cause the i 'th element of column one to be added to the i 'th element of column two and the result to be stored as the i 'th element of column three, where i takes on the values one through NRMAX, the number of rows of data in the columns. However, the second command will cause the constant 1 to be added to each element of column two and the results to be stored in column three.

There are two types of commands. Type I is the column-oriented command such as ADD or SUB. These commands perform certain operations using either constants or columns of values as indicated by the type of the arguments. The results are always stored in some column. Type II is the matrix-oriented command such as MADD or MSUB. These commands are generally distinguished from the column-oriented command by the prefix M. These commands either create matrices or, in various ways, manipulate arrays.

A further point needs to be made in regard to the column-oriented command. It should be noted that, in the graphics interactive version, each column contains 80 rows. As data are entered into the worksheet a counter keeps track of the number of rows used. When a column-oriented command is used, the operations are carried out down to the last row into which data have been entered.

Asterisks are recognized as special characters by OMNITAB and can be used in several ways. Three or more consecutive asterisks denote "through." For example, 1***5 would be interpreted as 1 2 3 4 5. Asterisks also provide the user with the means for using either data from the worksheet or any of five user defined variables, V, W, X, Y and Z, as arguments in commands. To use one of these variables or a worksheet entry as an argument it is necessary to enclose the variable or worksheet entry within asterisks. Single asterisks (*V* or *1,10*) are used to indicate a real argument while double asterisks (**V**, or **1,10**) indicate an integer argument. For example, suppose that the value of V is 1.5 and the element in the first row and tenth column of the worksheet is 2.6. The command:

ADD *V* *1,10* 3

would be equivalent to the command:

ADD 1.5 2.6 3

which would put the constant 4.1 into all previously used rows of column 3.

The command:

ADD **V** **1,10** 3

would be equivalent to the command:

ADD 1 2 3

Matrix commands begin with letter M. Matrices are referred to by the coordinates of the beginning argument in the worksheet, and by row and column size. For example MMULT 10 1, 5 BY 3; 16 1, 3 BY 4; 20 1 (', ', and BY are optional) directs the computer to take the matrix beginning at (Row 10, Col 1) ("coordinate 10,1") of the worksheet of dimension 5 BY 3, multiply it by a matrix starting in coordinate (16,1) (16,1), of dimension 3 by 4, and to store the results into locations beginning at coordinate (30,1).

Interactive OMNITAB is very easy to learn to use. This has been demonstrated repeatedly by students who have learned to use it following a brief demonstration. The opportunity to view results following execution of each command makes it easy for the user to find out what a command does. To illustrate this case, a description of portions of a typical demonstration session follows. In the process, many commands used frequently by the statistician will be illustrated.

When the user first sits down at the IBM 2250 console, he initially presses one of the lighted programmed function keys (PFK's) (see figure III-1). When the system requests a response, he will type \$LINK OMNITAB and the initial instruction frame will appear.

At this point several sections of the worksheet will be displayed when the user depresses lighted keys in the second and third row of the PFK keyboard.

Each worksheet section will contain an array of numbers. Next the command ERASE is entered. Then the worksheet sections viewed before are seen again. Now the worksheet sections contain nothing but zeros.

At this point one is ready to demonstrate the usage of some further commands. However, one first must enter data into the worksheet. One command which may be used is GENERATE. One might enter GENERATE 1.,.5,50.,1 . This command would generate values from 1. to 50. in steps of .5 into column 1. However an error message would appear indicating that the command exceeded the dimensions of the worksheet. (The command requires 99 rows but there are only 80 rows.) Next to be entered is GENERATE 1.,.5,25.,1. This command is executable. Following execution, PFK 4, which is used to display section 2 of the worksheet, is pressed and in column 1 can be seen the vector (1.0 1.5 2.0 ... 20.5). Pressing PFK 10 (to see section 7 of the worksheet) enables the user to see that the numbers 21. through 25. appear in rows 41 through 49 of column 1 of the worksheet.

The SET command can also be used to enter data into the one column of the worksheet as specified by the argument. The two lines

```
SET 2
1. 2. 3. 5. 9.
```

are entered. After pressing PFK 4, the user sees the vector (1. 2. 3. 5. 9.) in rows 1 through 5 of column 2. If the command SET 2 3 had been entered an error message would have stated that there were too many arguments since only one argument can be used with the SET command.

Now the user is ready to try some elementary operations using the data in these two columns. The user may wish to execute ADD 1 2 3 . By pressing PFK 4 he will see the vector (2. 3.5 5. 7.5 12.) in rows 1 through 5 of column 3. Rows 6 through 40 of columns 1 and 3 will be identical because the addition is carried out for rows 1 through 50.

If the user now decides that he wants to restrict his work to the first four rows, he enters the command RESET NRMAX 4. This has no immediate affect on the worksheet. However, if the next command entered is ADD 1 2 4, after pressing PFK 4, the user sees that rows 1 through 4 of column 4 now contain the values, 2., 3.5, 5. and 7.5. The remainder of column 4 is still filled with zeros.

Next one might enter the command ADD *4,1* 0.5 . This means that the number in coordinate (4,1) is added to 0. and the result stored into column 5. Pressing PFK 4 reveals that 2.5 now appears in rows 1 through 4 of column 5. These results can be seen in Figure II-2.

Finally consider the following series of commands:

READ 1 *** 4

The data to be typed will be entered, row after row, into columns 1 through 4 of the worksheet.

Display:	ROW 1
User:	1,2,4,8
Display:	ROW 2
User:	1,3,9,27
Display:	ROW 3
User:	1,44,16,64
Display:	ROW 4
User:	ROW 3 (he notices that he made an error in row 3)
Display:	ROW 3
User:	1,4,16,64
Display:	ROW 4 etc.

The 3 rows now appear in the first section of the worksheet (PFK 4)

DIVIDE 1 2 5

Entries in column 1 are divided by those in column 2, result is stored in column 5.

DIVIDE 1 2 6 5

This does the same thing as the previous instruction except that each quotient (1/2) is multiplied by the entries in column 6 before being stored in column 5.

WORKSHEET PART 1

COLUMNS

	1	2	3	4	5
1	1.00000	1.00000	2.00000	2.00000	2.50000
2	1.50000	2.00000	3.50000	3.50000	2.50000
3	2.00000	3.00000	5.00000	5.00000	2.50000
4	2.50000	5.00000	7.50000	7.50000	2.50000
5	3.00000	9.00000	12.0000	0.0	0.0
6	3.50000	0.0	3.50000	0.0	0.0
7	4.00000	0.0	4.00000	0.0	0.0
8	4.50000	0.0	4.50000	0.0	0.0
9	5.00000	0.0	5.00000	0.0	0.0
10	5.50000	0.0	5.50000	0.0	0.0
11	6.00000	0.0	6.00000	0.0	0.0
12	6.50000	0.0	6.50000	0.0	0.0
13	7.00000	0.0	7.00000	0.0	0.0
14	7.50000	0.0	7.50000	0.0	0.0
15	8.00000	0.0	8.00000	0.0	0.0
16	8.50000	0.0	8.50000	0.0	0.0
17	9.00000	0.0	9.00000	0.0	0.0
18	9.50000	0.0	9.50000	0.0	0.0
19	10.0000	0.0	10.0000	0.0	0.0
20	10.5000	0.0	10.5000	0.0	0.0

Figure II-2

Command Illustrations (partial display)

MIDENT 10 1 4 (or MIDENT 10 1 4 BY 4)

Generate an identity matrix, starting in coordinate (10,1), of order 4 by 4. This example illustrates the flexibility available in using the matrix commands. Since an identity matrix must be square, only one dimension needs to be specified. Both dimensions, however, may be used.

MDEFINE 15 1 4 4 1.

Generate a matrix of 1's, starting in coordinate (15,1), order 4 by 4.

MADD 10 1 4 4 15 1 15 1

Add the matrix starting in (10,1) (4 by 4) to that starting in (15,1) and store results back into the field starting at (15,1). If PFK 4 is pressed, the section now contains

	col 1	col 2	col 3	col 4
row 15	2	1	1	1
row 16	1	2	1	1
row 17	1	1	2	1
row 18	1	1	1	2

MINVERT 15,1 4 10 1

Invert the above matrix and store the inverse into the place where the identity matrix was, originally. Instead of READY, the usual message from the computer, it will display a message that the determinant is 5.

M(X'AX) 10 1 4 4 15 1 4 4 20 1

Take the matrix A from (10,1) on (4 by 4) and the matrix X from (15,1) on, (4 by 4), perform the $X'AX$ multiplication, and store results starting at (20,1).

GENERATE 1., .005, 1.095 1

Place the numbers 1., 1.005, 1.1, ..., 1.095 into the first 21 rows of column 1.

YORMX 1 2

Evaluate the normal c.d.f. for each of the entries in column 1 and store the result into column 2 (section of a normal distribution).

Conversely

GENERATE .90, .005, .995, 3

Generate numbers .90, .905, .91, ..., .995 in column 3.

YORMP 3 4

Take the percentage points of the normal curve corresponding to each entry in column 3, and place them into column 4.

The results of the last commands appear in Figure II-3.

These are but a few of the options available to the user. He may wish to study the OMNITAB manual but can press key 2 to see which commands are available in the interactive graphics version.

WORKSHEET PART 1

COLUMNS

	1	2	3	4	5
1	1.00000	0.841345	0.900000	1.28155	0.0
2	1.00500	0.842551	0.905000	1.31058	0.0
3	1.01000	0.843752	0.910000	1.34075	0.0
4	1.01500	0.844946	0.915000	1.37220	0.0
5	1.02000	0.846135	0.920000	1.40507	0.0
6	1.02500	0.847317	0.925000	1.43953	0.0
7	1.02999	0.848494	0.930000	1.47579	0.0
8	1.03499	0.849664	0.935000	1.51410	0.0
9	1.03999	0.850828	0.940000	1.55477	0.0
10	1.04499	0.851987	0.945000	1.59819	0.0
11	1.04999	0.853139	0.950000	1.64485	0.0
12	1.05499	0.854285	0.955000	1.69540	0.0
13	1.05999	0.855425	0.960000	1.75068	0.0
14	1.06499	0.856560	0.965000	1.81191	0.0
15	1.06999	0.857688	0.970000	1.88079	0.0
16	1.07499	0.858810	0.975000	1.95996	0.0
17	1.07999	0.859926	0.980000	2.05375	0.0
18	1.08499	0.861036	0.985000	2.17009	0.0
19	1.08998	0.862140	0.990000	2.32634	0.0
20	1.09500	0.863241	0.999000	2.57582	0.0

Figure II-3

Illustration of several statistical
commands (partial page)

CHAPTER III

IMPLEMENTATION

In this chapter, the implementation of an interactive OMNITAB version, for use by statisticians, is described in some detail. The first section contains a discussion of the problems and the approach to solutions. The second section contains detailed documentation.

A. PROBLEMS

At the University of Georgia the equipment used for interactive computer usage includes a graphics Console IBM 2250, attached to the IBM 360 Model 65. It consists of three parts: a Cathode Ray Tube Terminal which is used to display tabular and graphical information, a typewriter keyboard through which data and commands can be entered, and additional keys, referred to as "programmed function keys" (see Figure III-1), which are used by the user to branch to desired portions of the control program.

The IBM 2250 is operated under the control of a monitor (CMS[47]). One severe limitation is the restricted core available for this system (no more than 140K). The extensive programs of the interactive OMNITAB system had to be over-layed, and a structure had to be found which would minimize the number of calls from one overlay to another. We adopted the structure diagrammed in Figure III-2.

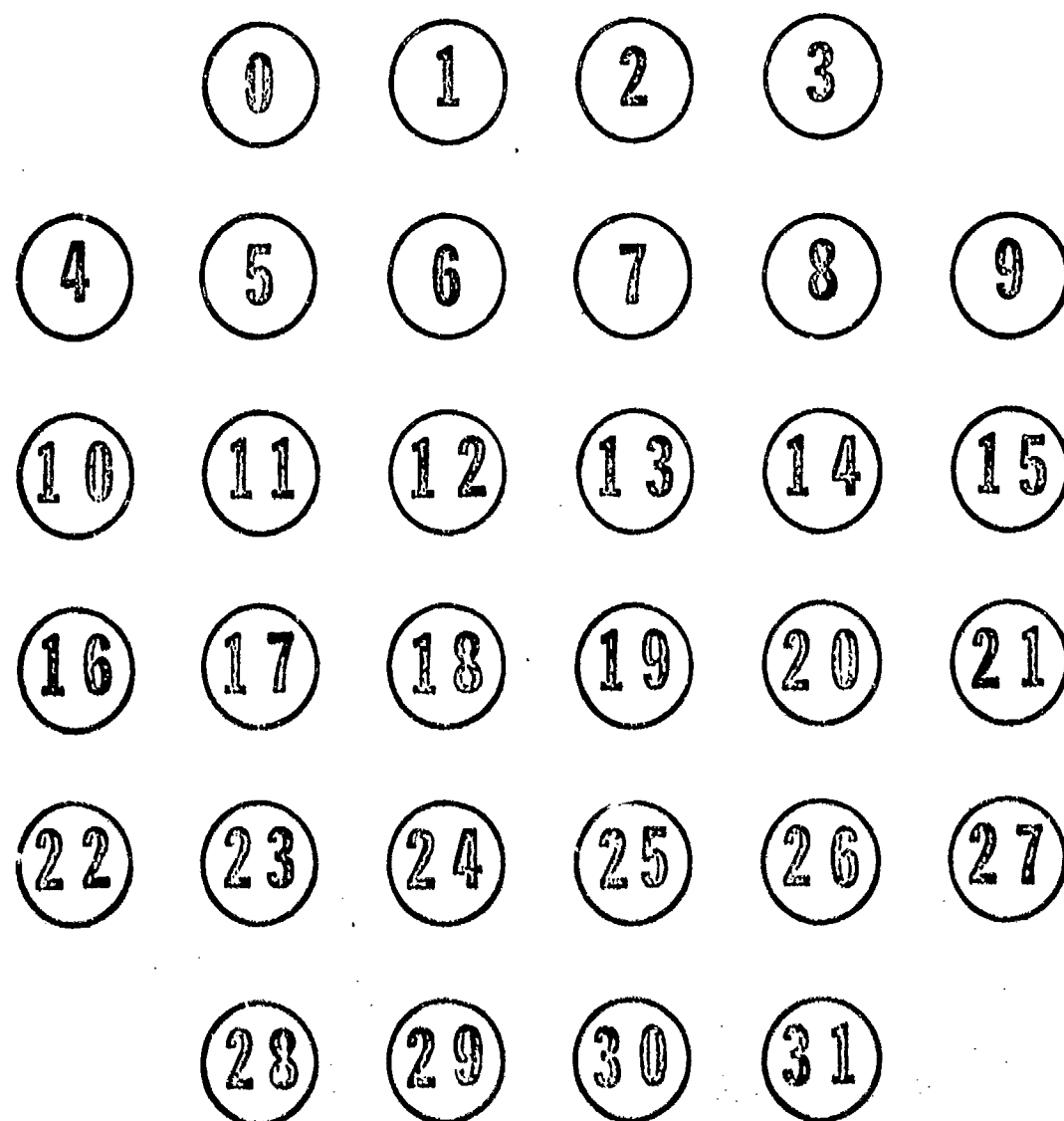


Figure III-1

Programmed Function Keyboard

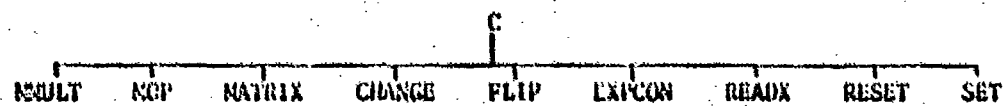
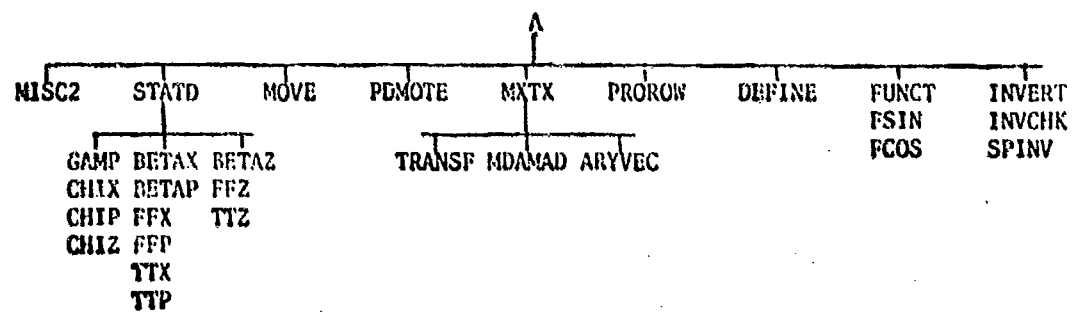
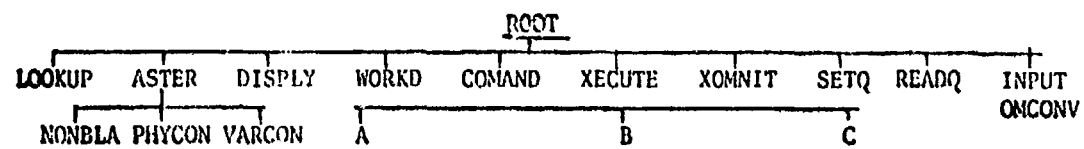


Figure III-2
Overlay Structure

The worksheet is the central feature of the interactive version of OMNITAB. A user should be able to view portions of it after each statement. A typewriter terminal would be too slow; even telephone-connected CRT terminals (e.g. Tektronix) would be unsatisfactory, since they perform display character-by-character. By contrast, a graphics terminal, such as the IBM 2250, presents whole page displays almost instantaneously, and is thus preferable for interactive worksheet systems. This nearly instantaneous display of the worksheet is one reason why this interactive implementation should appeal to the statistician.

The size of the worksheet presented a problem which required some consideration. In the original version of OMNITAB, the worksheet consisted of 101 rows and 46 columns. Later an option was added which enabled the user to reset the dimensions of the worksheet. In our conversational adaptation, variable dimensions were not used since programmed function keys were to be utilized to display desired sections of the worksheet; however, because of the core limitations, the worksheet for our interactive system was restricted to 80 rows and 30 columns. This is broken down into 12 sections of 40 rows and 5 columns each. The individual sections are addressable by 12 programmed function keys (rows 2 and 3 of the keyboard) in geometric analogy to the positions of these sections in the overall worksheet. Thus, a user may view any section of the worksheet by depressing the appropriate key.

The preparation of programs needed for conversation and display was facilitated by a set of systems routines COMFORT[47]. These subroutines provide an interface between a FORTRAN program and the 2250 display unit. Input and Output was dispatched by calls to these

subroutines wherever necessary in the controlling program.

Every programming language has a set of words and/or symbols which it recognizes as instructions. As part of the implementation of OMNITAB as an interactive language, it was necessary to consider what words would receive such special recognition. The command set for the batch-oriented OMNITAB consists of nearly two hundred such words. Many of these commands are not needed in an interactive language and consequently were not incorporated into the interactive OMNITAB vocabulary. Other commands are of no interest in solving statistical problems. They also were not incorporated. A number of routines which are important for statisticians have been added. These include evaluation of standard distribution functions (Normal, t, F, chi-square, Gamma, Beta) and matrix routines (determinant, trace). After these eliminations and additions, the OMNITAB command set for interactive use included about 125 commands. A list of these commands appears in Figure II-1 of Chapter II.

The general method of entering commands is illustrated in Figure III-3. The text (up to and excluding the word ERASE) is displayed to the user after he types \$LINK OMNITAB. - (ALT 5). The command (ERASE) is typed and entered (ALT-5)* by the user. The system responds to all commands with the two lines as shown in Figure III-3. If the user indicates that the command is correct he depresses Key 1, and it will then be executed. Otherwise he presses Key 2, and the command will be ignored and replaced by the next command which is entered. After execution of a command, the system will indicate that it is ready for the next command by responding READY as shown in Figure III-4. This

*In future discussions, "typing" will assume entering by the ALT-5 keys.

OUTPUT AREA

THIS PROGRAM IS DESIGNED TO ENABLE YOU TO USE OMNITAB COMMANDS ENTERED THROUGH THE TYPEWRITER KEYBOARD DIRECTLY IN FRONT OF YOU. TO SIGNAL COMPLETION OF YOUR COMMAND, FIRST DEPRESS THE "ALT" KEY, AND WHILE HOLDING IT DOWN, DEPRESS THE "5" KEY.

AT ANY TIME YOU MAY LOOK AT THE WORKSHEET BY PRESSING ANY OF TWELVE PROGRAMMED FUNCTION KEYS. EACH KEY WILL CAUSE A 40 BY 5 SECTION OF THE WORKSHEET TO BE DISPLAYED. KEYS 4 THROUGH 9 WILL DISPLAY THE FIRST 40 ROWS WITH KEY 4 DISPLAYING COLUMNS 1 THROUGH 5, KEY 5 DISPLAYING COLUMNS 6 THROUGH 10, ETC. KEYS 10 THROUGH 15 WILL LIKEWISE DISPLAY THE LAST 40 ROWS.

AFTER SEEING A PARTICULAR SECTION YOU MAY SEE ANOTHER SECTION BY PRESSING ANOTHER KEY OR YOU MAY ENTER MORE OMNITAB COMMANDS THROUGH THE TYPEWRITER KEYBOARD.

BY PRESSING KEY 30 YOU WILL RETURN TO THIS DISPLAY. BY PRESSING KEY 31 YOU WILL TERMINATE THIS PROGRAM. BY PRESSING KEY 3 YOU WILL BE ABLE TO SEE A DISPLAY OF THE OMNITAB COMMANDS CURRENTLY AVAILABLE. BY PRESSING KEY 2 YOU WILL SEE A LIST OF THE OMNITAB COMMANDS WHICH YOU HAVE ENTERED ERASE

THIS STATEMENT IS TECHNICALLY CORRECT. IF YOU WISH TO HAVE IT EXECUTED OR STORED, PRESS KEY 1. OTHERWISE, PRESS KEY 2.

REPLY AREA

Figure III-3

Entering a command after seeing the initial display

OUTPUT AREA

THIS PROGRAM IS DESIGNED TO ENABLE YOU TO USE OMNITAB COMMANDS ENTERED THROUGH THE TYPEWRITER KEYBOARD DIRECTLY IN FRONT OF YOU. TO SIGNAL COMPLETION OF YOUR COMMAND, FIRST DEPRESS THE "ALT" KEY, AND WHILE HOLDING IT DOWN, DEPRESS THE "5" KEY.

AT ANY TIME YOU MAY LOOK AT THE WORKSHEET BY PRESSING ANY OF TWELVE PROGRAMMED FUNCTION KEYS. EACH KEY WILL CAUSE A 40 BY 5 SECTION OF THE WORKSHEET TO BE DISPLAYED. KEYS 4 THROUGH 9 WILL DISPLAY THE FIRST 40 ROWS WITH KEY 4 DISPLAYING COLUMNS 1 THROUGH 5, KEY 5 DISPLAYING COLUMNS 6 THROUGH 10, ETC. KEYS 10 THROUGH 15 WILL LIKEWISE DISPLAY THE LAST 40 ROWS.

AFTER SEEING A PARTICULAR SECTION YOU MAY SEE ANOTHER SECTION BY PRESSING ANOTHER KEY OR YOU MAY ENTER MORE OMNITAB COMMANDS THROUGH THE TYPEWRITER KEYBOARD.

BY PRESSING KEY 30 YOU WILL RETURN TO THIS DISPLAY. BY PRESSING KEY 31 YOU WILL TERMINATE THIS PROGRAM. BY PRESSING KEY 3 YOU WILL BE ABLE TO SEE A DISPLAY OF THE OMNITAB COMMANDS CURRENTLY AVAILABLE. BY PRESSING KEY 2 YOU WILL SEE A LIST OF THE OMNITAB COMMANDS WHICH YOU HAVE ENTERED
ERASE
READY

REPLY AREA

Figure IVI-4

The screen when a command can be entered

procedure gives the user a chance to correct any typing errors which he may have made. The same procedure is followed for most commands.

As each command is entered and executed, it will appear at the bottom of the messages which already appear on the screen. The screen will only be erased when it becomes full or when one of the programmed function keys is used for special messages (PFK's 2, 3 and 30) or worksheet display (PFK's 4-15).

The READ command has several features which should be discussed. This command is used to enter data row by row into columns designated in the argument list of the READ command. In the batch version, data are usually available on cards or tape, in sequential order. For example, consider the statement

```
READ 1**3, 5, 7 .
```

The following cards would each contain one row of data to be placed into columns 1, 2, 3, 5 and 7 of the worksheet. The cards must be ordered by rows so that the first card in the sequence contains row one, etc. In our interactive environment, it must be possible to address any row at will. This was implemented as follows: When the user enters a READ statement the system responds, as shown in Figure III-5, by signalling that it is ready to receive row 1. This eliminates the annoying problem of pressing key 1 (as in other commands) after entering each line of data when many lines need to be entered. After data for Row 1 have been entered, the computer demands ROW 2(etc.). If the user discovers that he has made an error in entering data, he may veto the request by responding with ROW n where n is the number of that row at which he wishes to make the change. The next line of data

READ 1 2 3
ROW 1:

OUTPUT AREA

REPLY AREA

Figure III-5

Entering data using a READ command

is then entered in the designated row. The consequence of this command is the resetting of the row counter. The second line of data would enter the (n+1)st row unless the user indicates another row by using the ROW n command again. Exit from the read mode is accomplished by entering any other legitimate OMNITAB command.

B. DOCUMENTATION

In the implementation of interactive OMNITAB, the point of departure was the IBM 360 batch version developed by Chamberlain [15,37], since, for our purposes, it was more suitable than the original NBS version[31]. Those subroutines which required extensive changes, or even complete re-writing, have been indicated by an asterisk (*) in the summary beginning on page 35. Some entirely new subroutines, however, had to be written especially for interactive OMNITAB. Preceding the summary, the role played by each of those new subroutines is explained. In addition, some of the changes made in existing routines will be discussed.

As each command is entered through the keyboard, a number of checks is performed to determine if the arguments are legal. If the command appears to be executable, a subroutine PLBK is called. This subroutine performs several functions. First it writes the command on the screen and the message which follows as shown in Figure III-3. It then awaits the user's response. If the user replies by pressing key 1, the command will be written on a data set for later recall if desired and control will return to the subroutine where the command will be executed. If the user presses key 2, control will pass to the

driver subroutine where the user can correct his error and reenter the command.

The subroutine PRGRAM is used to display a listing of the commands which have been executed as shown in Figure III-6. It retrieves them from the data set on which the subroutine PLBK has written them. It also displays all the data which have been entered into the worksheet via READ commands.

The subroutine DISPLY is used to write the instructions which appear in Figure III-7. This frame is the first to appear when the user loads the OMNITAB package and briefly explains how to enter commands and what the programmed function keys will do.

The subroutine WORKD is used to present the different sections of the worksheet. Figure III-8 shows how a section is displayed. Other examples appear in chapter V.

The subroutine COMAND is used to list the commands currently implemented in this version of OMNITAB. If commands are added or taken out, this subroutine would have to be changed.

Finally the subroutine SCRAM is used when a large amount of scratch area which would not fit into the core region is needed; in this case, excess scratch space is provided on a disk as "virtual memory."

There were several types of changes from the IBM 360 batch version [37] that had to be made in this implementation. Each will be discussed in some detail. Many changes were required because this interactive version utilizes completely different input-output devices. A number of changes were made to enable the program to run in the limited amount of core available. Finally, several changes were made to achieve more efficient subprograms.

OUTPUT AREA
IF THE SCREEN BECOMES FULL AN ALARM WILL SOUND. WHEN YOU WANT TO SEE
THE NEXT SECTION OF YOUR PROGRAM, PRESS KEY 2.

ERASE
READ 1 2 3
1 2 3
2 4 6
4 2 5
8 9 1.5
ROW 5:

REPLY AREA

Figure III-6

Listing of commands entered and recovered using PROGRAM

OUTPUT AREA

THIS PROGRAM IS DESIGNED TO ENABLE YOU TO USE OMNITAB COMMANDS ENTERED THROUGH THE TYPEWRITER KEYBOARD DIRECTLY IN FRONT OF YOU. TO SIGNAL COMPLETION OF YOUR COMMAND, FIRST DEPRESS THE "ALT" KEY, AND WHILE HOLDING IT DOWN, DEPRESS THE "S" KEY.

AT ANY TIME YOU MAY LOOK AT THE WORKSHEET BY PRESSING ANY OF TWELVE PROGRAMMED FUNCTION KEYS. EACH KEY WILL CAUSE A 40 BY 5 SECTION OF THE WORKSHEET TO BE DISPLAYED. KEYS 4 THROUGH 9 WILL DISPLAY THE FIRST 40 ROWS WITH KEY 4 DISPLAYING COLUMNS 1 THROUGH 5, KEY 5 DISPLAYING COLUMNS 6 THROUGH 10, ETC. KEYS 10 THROUGH 15 WILL LIKEWISE DISPLAY THE LAST 40 ROWS.

AFTER SEEING A PARTICULAR SECTION YOU MAY SEE ANOTHER SECTION BY PRESSING ANOTHER KEY OR YOU MAY ENTER MORE OMNITAB COMMANDS THROUGH THE TYPEWRITER KEYBOARD.

BY PRESSING KEY 30 YOU WILL RETURN TO THIS DISPLAY. BY PRESSING KEY 31 YOU WILL TERMINATE THIS PROGRAM. BY PRESSING KEY 3 YOU WILL BE ABLE TO SEE A DISPLAY OF THE OMNITAB COMMANDS CURRENTLY AVAILABLE. BY PRESSING KEY 2 YOU WILL SEE A LIST OF THE OMNITAB COMMANDS WHICH YOU HAVE ENTERED READY

REPLY AREA

Figure III-7

Basic Instructions

For the batch version of OMNITAB, the input device used is generally a card reader. The OMNITAB commands and data are punched onto cards. The OMNITAB compiler reads each card and executes the requested task sequentially. For the interactive version of OMNITAB, the commands are entered from the IBM 2250 keyboard. Each instruction is executed as it is entered. The user can not enter a command until the previous one has been executed. In addition, the user may exercise several options following the execution of any command.

Changes in several subprograms had to be made so that the above procedure could be used. In the batch version of OMNITAB input was achieved with a call to the subroutine INPUT. In the interactive version, this one CALL statement was replaced by thirty-eight statements beginning at label 52. (See Appendix p. 259) This sequence of statements performs the various tasks associated with input. This includes a sequence to write out the line READY to inform the user that another command may be entered when this is appropriate. It also includes a sequence which writes out the row number when the user is entering data. A number of statements are required to handle the interrupts received from the programmed function keys. Finally a body of code was necessary to control the flow of the program through these various options. In the subroutine INPUT it was also necessary to replace the READ statement by a call to the subroutine GRNPLY which retrieves a line entered from the keyboard.

In the batch version of OMNITAB, output was dispatched to a printer. There were a number of commands which could be used for obtaining the desired output from the program. In the interactive version all of these commands had to be eliminated. This meant that

references to these commands in LOOKUP and OMNIT had to be eliminated. On the other hand it was necessary to make provision for retrieving intermediate results from the worksheet. The subroutine WORKD, discussed in the beginning of this section was written to satisfy this need.

Error messages are related to the problem of output. The two subroutines ERROR and AERR required extensive modification. In both subroutines messages no longer necessary were removed and new messages had to be provided. In both subroutines WRITE statements could not be used. They were replaced by calls to the subroutine GRDPLY which writes lines on the screen. In this interactive version only one error message can be displayed for each command. A check, therefore, had to be displayed for each command. A check, therefore, had to be inserted at the beginning of ERROR to prevent multiple messages from being displayed. For fatal errors this flag will prevent execution until a correction is made. For arithmetic errors, the flag will prevent recording of the command on the temporary data set used to store a record of the executed program. This is generally done following execution of a command. Thus it was necessary to insert several statements into ERROR to record the command onto the data set at this point. For informative diagnostics, the user may choose to override the flag and have the command executed. Several more statements had to be added to ERROR to provide the user with this option.

The core limitation has been mentioned previously in connection with the worksheet size. Because of this limitation, a number of changes had to be made in several subprograms. In the batch version of OMNITAB a fairly large scratch area was available in core for

storing intermediate results for such commands as matrix inversion. In the interactive version a small scratch area capable of holding a full column of data was retained in core and the subroutine SCRAM was written, which extends the scratch area to disk, for use when a larger area is needed. Those subroutines which had to be extensively modified for this reason are INVCHK, INVERT, MATRIX, MDAMAD, MMULT, MOVE, MXTX, SPINV and TRANSF.

The subroutines INVCHK, INVERT and SPINV are used for matrix inversion. The problems caused by a lack of scratch area were most pronounced in these subprograms. In INVCHK, an error bound was calculated and supplied to the user whenever he inverted a matrix. In the interactive version, the determinant of a matrix was displayed instead, as statisticians need it from time to time and precision can be easily checked by reinversion when necessary. As a consequence, the section in INVCHK which calculated the error bound was removed and a number of statements were added to SPINV to calculate the determinant.

Several examples of changes made to achieve a greater efficiency will be discussed here. It should be pointed out that the developer's of OMNITAB made an effort to use algorithms which would provide the user with a relatively high degree of precision. An example of this is the use of Walsh's orthonormalizing algorithm, ORTHO[3,19,20,21,61]. This provided a very good matrix inversion routine [43,62,63]. For this reason, minor changes only were made.

In the subroutines ARITH and FUNCT, several computed GO TO statements were used to repeatedly branch to the section of the program where a given operation is performed. These computed GO TO statements

were replaced by assigned GO TO statements. Several other changes, such as the insertion of ASSIGN statements, were associated with this change.

In the subroutine TRANSF which is used for the matrix operations XAX' and $X'AX$, the multiplication is performed in one step. A much more efficient method is to perform two matrix multiplication steps. This required rewriting an entire section of TRANSF. The same section also had to be changed because of the reduction in scratch area discussed earlier.

Two other subroutines which required extensive modification were LOOKUP and XECUTE. LOOKUP determines if a given command is legitimate. Since many commands were added and others were removed, changes needed to be made to reflect this. Similarly, XECUTE, which is used to pass control to the subroutine appropriate to a given command, had to be changed to reflect the additions to and deletions from the set of commands.

The documentation which appears at the end of this chapter provides information concerning all subprograms which have been modified in any way. Since there does not seem to be detailed documentation of these programs in their original version [5,13], the description given in this section is as self-contained as possible, and not restricted to a description of changes necessary for adaptation to interactive statistical use. The information provided includes a statement of purpose of each routine, the COMMON block variables used, and a brief trace of the program logic. It should be read in conjunction with the listing of the routine (See Appendix.). It is hoped that this description will prove helpful in making similar adaptations elsewhere. Documentation of the statistical distribution subprograms can be found in Bouver [5]. Since

the documentation is arranged alphabetically, it is desirable first to describe briefly the purposes of the documented subprograms in accordance with the various types of functions which the subprograms serve.

There is a very short MAIN* program which opens and closes files, initializes the Graphics Monitor System and calls OMNIT*. OMNIT* is the principal subroutine and controls execution of the user's commands. The subroutine XOMNIT*, called by OMNIT*, initializes several variables to values which indicate the beginning of a user's session.

A number of subprograms are used for translating the user's command into a form which can be used for execution of the command; (this is discussed in greater detail in Chapter IV); INPUT* picks up the line entered in the reply area of the CRT. OMCONV* converts the entered character string into a numerical code. NNAME converts the keyword into two numerical values which uniquely identify the command. NONBLA is used in scanning the line to find non-blank characters. AARGS* is used to convert a string of digits into the appropriate number. Whenever asterisks are encountered, ASTER is used to obtain information needed to finally obtain the indicated argument(s). ASTER will call PHYCON* and VARCON if a name appears within asterisks. These subroutines identify legitimate names as either physical constants or variables. EXPAND performs the final translation using XPND for arguments which used asterisks.

After translation is successfully completed several more subroutines are needed before execution can begin. LOOKUP* checks to see if the entered commands are legitimate. XECUTE* calls the subroutines in which execution of the various commands is actually carried out.

A number of subroutines are available for use in subroutines which execute commands. ADRESS calculates the address of a desired argument. CHKCOL checks to see if all arguments are legitimate column numbers and, if so, obtains the addresses of the columns in the worksheet array. CKIND checks to see whether the arguments are all floating point, all integer, or mixed. MTXCHK checks to see if matrices fit in the worksheet and calculates their addresses in the worksheet array. VECTOR stores a single constant in an entire column. SCRAM is used when extensive scratch space is needed during execution of a command. PLBK is used to halt execution while the user checks the command which he has entered.

Two subroutines are used for displaying error messages. ERROR* displays messages when the error is syntax. AERR* displays messages when the error is arithmetic.

ECBINT and PFINT set a variable ITYPE to indicate the type of interrupt. If the interrupt was a programmed function key interrupt, then several subroutines are called to respond to the user's direction. COMMAND displays the list of implemented commands. DISPLY produces the display which appears initially and contains some instructions. PROGRAM displays the user's entered program. WORKD retrieves and displays sections of the worksheet array.

Several subroutines are used to enter data into the worksheet. GENER* is used to generate a column of numbers by specifying first number, last number and increment. READX* performs initialization necessary when entering data following a READ command. READQ* is used to enter rows of data following a READ command. SET* performs

initialization necessary when entering data following a SET command.

SETQ* is used for entering data following a SET command.

Finally, there are many subroutines which are used for computation in response to specific commands. These are the following: ARITH*, ARYVEC, CHANGE*, DEFINE, ERASE, EXCHNG, EXPCON*, EXTREM, FLIP, FUNCT*, FCOS, FEXP, FEXP2, FLOG, FSIN, FSQRT, INVERT* (which uses INVCHK* and SPINV*), MATRIX*, MDAMAD*, MISC2*, MMULT*, MOP*, MOVE*, MRAISE, MSCROW, MXTX*, PDMOTE, PROROW, RESET, SORDER, STATD and TRANSF*.

THE COMMON BLOCKS

<u>COMMON BLOCK</u>	<u>VARIABLE NAME</u>	<u>DESCRIPTION</u>
BLOCKA	MODE=1	For interactive mode
	MODE=2	For input mode.
	M	A pointer and scans through the array KARD.
	KARD(77)	An array which contains a numerical representation of the input line.
	KARG, ARG ARG2	Used for various purposes during the compilation of the argument portion of an input line.
	NEWCD(19)	Contains the most recent input line.
	KRDFND	Set to the maximum number of characters in an input line.
	NEWCDS(19,5)	Used to save 5 consecutive input lines before writing them out on a data set.
	KSAVE	Contains the number of lines in NEWCDS which have not been written out.
	NSAVE	The data set number used for both storing input lines and as a scratch area.
	NFLAG	Used to prevent execution of a command. Normally NFLAG is 0. NFLAG is set to 1 to prevent execution.

<u>COMMON BLOCK</u>	<u>VARIABLE NAME</u>	<u>DESCRIPTION</u>
BLOCKD	RC(2439)	Contains the worksheet (2400 elements) and the floating point argument list (39 elements).
	IARGS(69)	The integer argument array.
	KIND(39)	An array used to determine whether the i'th argument is floating point (KIND(I)=1) or integer (KIND(I)=0).
	ARGTAB(51)	Contains information obtained in the sweep of the input line and is used to obtain the arguments for a command.
	NRMAX	The number of rows being used.
	NROW,NCOL	The number of rows and number of columns of the worksheet.
	NARGS	Generally contains the number of arguments in the input line but is modified during execution of some commands.
	VWXYZ(5)	The user's variable array.
BLOCKE	NAME(4)	An array containing the numerical representation of the command and any command modifier.
	L1	Indicates the group to which a command belongs.

<u>COMMON BLOCK</u>	<u>VARIABLE NAME</u>	<u>DESCRIPTION</u>
BLOCKE	L2	Indicates which command within a group has been used.
	ISRFLG=0	For READ initiated input.
	ISRFLG=1	For SET initiated input.
BLOCKF	NCTOP	Contains the top row of the worksheet and is always one.
CONSTS	PI, E and HALFPI	Contains the values of π , e and $\pi/2$ for internal use by the program.
	DEG and RAD	Are used for converting from degrees to radians and from radians to degrees.
KPLOT		The variables in KPLOT are used in obtaining CALCOMP plots of the screen.
PCONST	P	An array containing the values π and e.
	N	An array containing the numerical translation (OMNITAB code) of the characters PI and E. This enables the user to type PI and E and have the program recognize them as constants.
QRS	NDROW	Marks the end of the column into which data are placed using the SET command
	J	Is used with both READ and SET to indicate.

<u>COMMON BLOCK</u>	<u>VARIABLE NAME</u>	<u>DESCRIPTION</u>
	J (cont.)	the row into which data are to be placed.
	NNARG	Used in the READ initiated input mode and contains the number of arguments in the READ command.
SCRAT	A	Used as a scratch area.

MAIN PROGRAM AND SUBPROGRAMS

NOTE: Throughout this section, the word "coordinate" refers to either the "row, column" designation of the upper left hand corner of a matrix or to the position in the worksheet array RC where the upper left hand element of the matrix is located.

AARGS

PURPOSE: This subroutine reads a string of digits and assembles a number from them.

COMMON BLOCK

BLOCKA

VARIABLES USED

M, KARD, KARG, ARG

LINE
NUMBERFORTTRAN
LABELCOMMENT

12

ARG starts out as the first number found.

13,26,27

10,110

SIG contains the sign of the number

23,41

40

KARG=1, number is in floating point mode.

17

KARG=0 until a decimal point is found.

18,30

KEXP contains the number of digits.

65,66

A number containing more than 10 digits has to
be real.

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENT</u>
16,24,31		The value of IEXP indicates how many digits lie to the right of the decimal point.
27-33	20	The value of the number is obtained as ARG.
38-40		Error: Two decimal points were found.
15,54		IXS contains the sign of the exponent.
46,53	50,52,54	Check for exponent following the number.
56	56	A number with an exponent must be real.
14,57	70	JEXP contains the value of the exponent.
72,73,75	123,126	Multiply or divide by the appropriate power of 10.
67	110	Attach the appropriate sign.
77	130	Error: Real number is too large to store in the machine.

ADRESS(I,J)

PURPOSE: This subroutine calculates the addresss of argument I. If the argument is a legal column number, J will be equal to the location of the top of the column in the worksheet array RC. If the argument is an illegal column number, J will be 0. If the argument is a floating point number -J will be the address of the argument in the worksheet array RC. The end of the array RC (elements 2401-2439) is used to store floating point arguments.

COMMON BLOCK

BLOCKD

BLOCKF

VARIABLES USED

RC, IARGS, KIND, NROW, NCOL

NCTOP

LINE NUMBER

FORTRAN LABEL

COMMENT

14		Calculate the location of a roal argument in the array RC.
16	10	Check to see if column number is legal.
19	20	Calculate the beginning of the column in RC.

AERR(I)

PURPOSE: This subroutine causes error messages to be written on the screen for arithmetic errors only. I is the error code.

AERR(I) (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENT</u>
8		I determines which message is to be written.
9-23	55,201- 207	Error messages.

ARITH

PURPOSE; This subroutine is used to execute the commands ADD, SUB, MULT, DIV, RAISE, SUBTRACT, MULTIPLY and DIVIDE.

<u>COMMON BLOCK</u>	<u>VARIABLES USED</u>
BLOCKA	NFLAG
BLOCKD	RC, KIND, NRMAX, NARGS
BLOCKE	L2 = 1 for ADD; = 2 for SUB = 3 for MULT; = 4 for DIV = 5 for RAISE; = 6 for SUBTRACT = 7 for MULTIPLY; = 8 for DIVIDE

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENT</u>
13		Equivalence of user commands: SUBTRACT=SUB, MULTIPLY=MULT, DIVIDE=DIV.
14,17,25, 20	2	Check for various errors.
22-28	15,20,30	Obtain address of column or location of constant for each argument.

ARITH (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENT</u>
34		JJ marks the end of the column into which the results are to be stored.
35,36		IJ is used to determine which loop to use to execute a given command.
37-95		There is a DO-loop for each type of command.

ARYVEC

PURPOSE: This subroutine is used to execute the commands M(AV) and N(V'A).

<u>COMMON BLOCK</u>	<u>VARIABLES USED</u>
BLOCKA	NFLAG
BLOCKD	RC, IARGS, NARGS
SCRAT	A
BLOCKB	L2 = 6 for M(AV) =7 for N(V'A)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENT</u>
15		There must be either 6 or 7 arguments.
19-21		All arguments must be integers.
26-27		The fifth argument must contain the column number of the vector.

ARYVEC (ccnt.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENT</u>
31-32		For the command V'A, ICS will contain the row in which the result is to be stored (and must be within the worksheet).
34	440	For AV, ICS will contain the beginning of the storage column in the worksheet.
37-41	450	For the 7-argument case, treat the result as an a by 1 or 1 by b matrix. Note that L2=6 for AV and L2=7 for V'A.
43,44	460	Check the legality of the matrix or matrices.
51		IAP contains the address of A.
52		For the 7-argument case, set ICS to the coordi- nate of the output matrix.
53		IP contains the length of the resulting vector.
55-61	640	JP contains the implied length of V. IAD1 and IAD2 are increments used for obtaining the correct result in the multiplication. They are set to a, 1 for AV and to 1, b for V'A.
62-71	660-740	Perform the multiplication using the scratch array A to hold the resulting vector.
75-76		Store the resulting vector into the designated location in the worksheet.

ASTER

PURPOSE: This subroutine is used when asterisks are encountered in an argument list. It compiles that part of the argument list which uses asterisks.

COMMON BLOCK

BLOCKA

VARIABLES USED

M, KARD, ARG, ARG2

KARG = 1	Single Asterisks	} as input
= 0	Double Asterisks	
= 1	Error	} as output
= 2	Floating Point Constant	
= 3	Integer Variable	
= 4	Floating Point Variable	
= 5	Worksheet Entry, to be used as an integer	
= 6	Worksheet Entry, to be used as a floating point number	
= 7	Asterisks, indicating through	

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
20	10	Check for string of asterisks.
24,46,59 70,80	80,110	KARG indicates how the asterisks were used.
25,26	15	*** to mean "through."
28	20	Error: Number or letter must follow asterisks.
29		Jump if a letter is found
34-40	30,40,45	Establish a worksheet reference.

ASTER (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
34	30	Determine the row or column number.
35		Error: Argument must be integer.
36		Row number must be followed by comma and column number must be followed by asterisk.
39		Error: No column number.
44-47	45	ARG=row number; ARG2 = Column number.
53	50	Code the name.
54-55		Check to see if name is a physical constant.
60		Error: Physical constant must be real.
65-66	60	Check to see if name is a logical variable name.
67		Error: The name is not legal.
68	70	KARG=1 indicates an error.
71-76	90,100	Error: Number of asterisks following expression is not equal to the number preceding the expression.

BLOCK DATA

PURPOSE: This subprogram initializes some constants.

COMMON BLOCK

BLOCKA

BLOCKD

BLOCKC

VARIABLES USED

MODE, KRDEND, KSAVE, NSAVE, NFLAG

NIMAX, NROW, NCOL

L1

BLOCK DATA (cont.)

COMMON BLOCKVARIABLES USED

BLOCKF

NCTOP

CONSTS

PI, E, HALFPI, DEG, RAD

KPLOT

NFRAME, KKND, SIZE, SPACE

PCONST

P, N

LINE
NUMBERFORTRAN
LABELCOMMENTS

15,16

Set some constants used in the system.

17

P contains the values of π and e and N contains
the OMNITAB code of the characters PI and E.

19

NROW and NCOL contain the dimensions of the
worksheet. NFRAME, KKND, SIZE, SPACE are
used for preparing CALCOMP plots.

CHANGE

PURPOSE: This subroutine executes the command CHANGECOMMON BLOCKVARIABLES USED

BLOCKD

RC, NRMAX, NARGS

BLOCKA

NFLAG

LINE
NUMBERFORTRAN
LABELCOMMENTS8-11,
20-24903,910
909

Check for errors

15

J will be the beginning of the column.

CHANGE (cont.)

17,18 20 Change signs.

CHKCOL(J)

PURPOSE: This subroutine checks to see if the first NARGS integer arguments are legitimate column numbers. J will be 0 for no error and 1 for error. IARGS(1) through IARGS(NARGS) will become the addresses of the columns in the worksheet array RC.

COMMON BLOCKVARIABLES USED

BLOCKD

IARGS, NARGS

LINE
NUMBER

FORTRAN
LABEL

COMMENT

11

10

J is set to 1 if any error is found.

10

There must be at least one argument.

13-15

20

Check to see if each argument is a legal column number.

17

J is set to 0 if no error is found.

CKIND(J)

PURPOSE: This subroutine checks the type of the first J arguments. J is set to 0 if all are integers, to 1 if all are floating point numbers, and to 2 if both types are found.

COMMON BLOCKVARIABLES USED

BLOCKD

KIND

CKIND(J) (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENT</u>
9		JA will be the number of arguments to be checked.
10		J will remain 0 if no floating point numbers are found.
11-13	10	Check for floating point numbers.
15	15	J is 1 if no integers are found.
16-18	20	Check for integers.
20	30	J is 2 if both types are found.

COMAND(*)

PURPOSE: This subroutine displays on the screen a list of available
commands.

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENT</u>
8-35		Those two data statements contain the names of all commands currently implemented.
37		NSIZE is the number of commands. The array NAMES must be dimensioned 2* NSIZE.
38		NROWS contains the number of full rows of commands which will be displayed on the screen.
39		NLEFT contains the number of commands to be displayed in the last row of commands.

COMAND(*) (cont.)

40		Erase the screen.
41-42		Write a heading.
43-48	50-70	The array K contains 0's and 1's. The first NLEFT elements are 1 and the remainder are 0. This array is used in obtaining from the array NAMES the names which belong in a given row.
50-60	100-200	Write out each full row of commands. The array NWORK is used to hold the names pulled out from the array NAMES for each row.
62-70	300	The last line may not be a full line.

DEFINE

PURPOSE: This subroutine is used to execute the command DEFINE.

COMMON BLOCK

BLOCKA

BLOCKD

BLOCKE

VARIABLES USED

NFLAG

RC, IARGS, KIND, NRMAL, NARGS, NROW

L1, L2, J

LINE
NUMBERFORTRAN
LABELCOMMENT

13

Check for illegal number of arguments.

24

10

K1=0 for integer, 1 for real.

25

Error: The first argument in the 4-argument
form is real.

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
27		L2=2 (column, column) =3 (row, column, column) and (constant, column) =4 (row, column, row, column) and (constant, row, column).
30		L marks the beginning of the storage column.
31		Error: Last argument must be a column number.
32	20	Error: Only when L2=4 can NRMAX be 0.
34-37	30	When L2=4, L must be changed to point to the requested row.
39-42	50,55	J marks the beginning of the origin column.
43-46		J is changed to point to the requested row.
47		ARGS(1) must contain the required constant.
51-54	65,70	Copies first column into second.
56	80	The required value is copied into the first NRMAX rows of the referenced column.
58	90	The required value is copied into the designated row and column.

DISPLY(*)

PURPOSE: This subroutine is used to produce the initial display
which appears after the user enters \$LINKONTAB.

EOBINT

PURPOSE: This subroutine sets ITYPE to 2 when the wait state is interrupted by an EOB.

COMMON BLOCK

BLANK

VARIABLES USED

ITYPE

ERASE

PURPOSE: This subroutine is used to execute the command ERASE.

COMMON BLOCK

BLOCKA

BLOCKD

VARIABLES USED

NFLAG

IARGS, NRMAX, NROW, NCOL, NARGS

LINE
NUMBERFORTRAN
LABELCOMMENTS

9,10

All arguments must be column numbers (I=0)

17,18

40

Zero out each column in the argument list.

23,24

50

Erase the entire worksheet.

ERROR(I)

PURPOSE: This subroutine is used whenever an error is detected, and informs the user as to the nature of the error.

COMMON BLOCK

BLANK

VARIABLES USED

IOVLY, KEY

ERROR(I) (cont.)

COMMON BLOCK

KPLOT

BLOCKA

VARIABLES USED

NFRAME, KKND, SIZE, SPACE

NEWCD, NEWCDS, KSAVE, NSAVE, NFLAG

LINE
NUMBERFORTTRAN
LABELCOMMENTS

10

For arithmetic errors, this prevents the
error message from being repeatedly
displayed.

11,12

J will be zero for arithmetic errors.

15

Display the entered command.

17

J will be 1 for informative diagnostics.

18

NFLAG=1 indicates error condition.

19-68

1-30,3000,
5000

Display error message for fatal errors.

77-88

9001,9004,
250,9003

Since an arithmetic error has occurred
the command must be written on a data
set here rather than after execution
has been completed.

92-116

400-415,
4000,4500

Display informative diagnostics.

117-120

PPK22 will initiate preparation of a data
set for plotting the CRT image on a
CALCOMP plotter.

EXCHNG

PURPOSE: This subroutine is used to execute the command EXCHANGE

COMMON BLOCK

BLOCKD

BLOCKA

VARIABLES USED

RC, IARGS, NRMAX, NARGS

NFLAG

LINE
NUMBERFORTTRAN
LABELCOMMENTS

7

There must be an even number of arguments.

8,9

All arguments must be column numbers.

13-20

30

The exchange command is executed.

EXPAND(J,WHERE)

PURPOSE: This subroutine is used to translate information which has been stored in the array WHERE into a form which will be used for the actual execution of commands. J indicates the number of elements in WHERE containing information needed for the current command.

COMMON BLOCK

BLOCKD

VARIABLES USEDIARGS, ARGS(equivalenced to RC), KIND,
NARGSLINE
NUMBERFORTTRAN
LABELCOMMENTS

12

II will be used as the subscript for the arrays ARGS, KIND and IARGS.

EXPAND(J,WHERE) (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
13		I will be used as the subscript for the input array WHERE.
14		JJJ marks the end of the array WHERE.
15-17	10,15	Increment the subscripts and check to see if any more conversion is necessary.
20-22	20	Positive T indicates an integer argument. Set KIND to 0 and store the argument into IARGS.
23-26	30	If T is 0 then the next element of WHERE contains the next argument which is real. Set KIND to 1 and store the argument in ARGS.
28-36	41-50	The subroutine XPND obtains arguments when reference is made using a variable or a worksheet location.
40	100	The following section is used when *** was found in the command.
43		Error: Both arguments surrounding *** must be integers.
45	105	IU will contain the integer argument following ***.

EXPAND(J,WHERE) (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
46-52	106-125	Increase NARGS to include the number of implied arguments and check to determine whether expansion is from a high value to a lower value or from a low value to a higher one.
53-56	140,150	Fill IARGS with the implied arguments and set the corresponding elements of KIND to 0.
58-65		Use the subroutine XFND to obtain the value of the integer argument following III.

EXPLOX

PURPOSE: This subroutine is used for executing the commands
MVECDIAG, AVECDIAG, MVECMAT, AVECMAT, FMATVEC and AARRVEC.

COMMON BLOCK

BLOCKA

BLOCKD

BLOCKE

VARIABLES USED

NFLAG

RC, IARGS, NROW, NARGS

L2 = 1 MVECDIAG

= 2 AVECDIAG

= 3 AVECMAT

= 4 FMATVEC

EXPCON (cont.)

COMMON BLOCKVARIABLES USED

BLOCKE (cont.)

= 5 AARRVEC

SCRAT

A

LINE
NUMBERFORTTRAN
LABELCOMMENTS

10

There must be either 5 or 6 arguments.

12-14

100

All arguments must be integers.

15-17

The first 4 arguments identify the matrix
Check for errors.

18,19

102

ILL will contain the address of the
column in the command.

20-24

Initialize constants that will be used in
performing the required operation.

25

For MVECMAT, AVECARR, MMATVEC and AARRVEC,
the implied length of the vector is
the product of the number of rows and
number of columns of the matrix. This
must be limited to 80, the length of a
column.

26-28

If there are 6 arguments, ILC, the address
of the vector, must be adjusted to a
row other than the first row of a
column. The implied length of the
vector is further restricted also..

29

103

IXX marks the end of the vector.

EXPCON (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
36-43	120-130	The diagonal of the matrix is stored in the indicated column.
45-58	220-250	The designated matrix is stored as a vector in the designated column.
60-70	300-305	The arguments are moved so that the first four arguments designate the matrix.
71-82	310-340	The designated vector is stored as a matrix.

EXTREM

PURPOSE: This subroutine is used for executing the commands
MAX, MAXIMUM, MIN and MINIMUM.

COMMON BLOCK

BLOCKD

BLOCKA

BLOCKB

VARIABLES USED

RC, IARGS, NCMAX, NARGS

NFLAG

L2 = 4 MAX

= 5 MAXIMUM

= 6 MIN

= 7 MINIMUM

EXTREM (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
10		There must be an even number of arguments.
14,15	30	All arguments must be column numbers.
23,24		J is used to determine where the maximum or minimum is in the column. If NRMAX is 1, then it has to be in the first row.
25-27		Prepare constants for use in the search.
32-35	70	Find the maximum.
39-41	80,90	Find the minimum.
42	100	J will be one less than the number of the row containing either the maximum or the minimum.
43-45	110,120	Execute the command.

FCOS(X)

PURPOSE: This function evaluates the cosine of x, checking first to determine that the value of x is within the bounds of the function and returning 0 if the cosine cannot be evaluated.

FEXP(X)

PURPOSE: This function evaluates e^x , checking first to see if overflow would result and returning 0 if overflow or underflow

FEXP(X) (cont.)

would occur.

FEXP2(B,E)

PURPOSE: This function evaluates B^E if possible and returns 0 if the evaluation would produce underflow or overflow or if B is negative.

FLIP

PURPOSE: This subroutine is used for executing the command FLIP.

COMMON BLOCK

BLOCKA

BLOCKD

VARIABLES USED

NFLAG

RC, IARGS, NRMAX, NARGS

LINE
NUMBER

FORTTRAN
LABEL

COMMENTS

7

There must be an even number of arguments.

11,12

20

All arguments must be column numbers.

20-35

50,60

Flip column IARG(I) into column IARG(I+1)
for each argument pair.

FLOG(X)

PURPOSE: This function evaluates $\ln x$ for x greater than 0. It returns 0 for x less than or equal to 0.

FSIN(X)

PURPOSE: This function evaluates $\sin x$ if x is within the bounds of the function. It returns 0 if x is not.

FSQRT(X)

PURPOSE: This function evaluates the square root of non-negative x . It returns 0 if x is negative.

FUNCT

PURPOSE: This subroutine is used for the execution of the following commands: SIN, COS, TAN, COT, ARCSIN, ASIN, ARCCOS, ACOS, ARCTAN, ATAN, ARCCOT, ACOT, SIND, COSD, TAND, COTD, ASIND, ACOSD, ATAND, ACOTD, ABS, ABSOLUTE, EXP, EXPONENT, LOG, LOGE, SQRT, NEGEXP, LOGTEN, ANTILOG, SINH, COSH, TANH, COTH, ASINH, ACOSH, ATANH, ACOTH and DEVNOR.

COMMON BLOCKVARIABLES USED

BLOCKA

NFLAG

BLOCKD

RC, KIND, NRMAX, NARGS

CONSTS

HALFPI, DEG, RAD

BLOCKE

L2 = 1 ABS; 2 EXP

= 3 LOG; 4 SQRT

= 5 NEGEXP; 6 LOGTEN

= 7 ANTILOG; 8 SINH

= 9 COSH; 10 TANH

FUNCT (cont.)

COMMON BLOCK

BLOCKE (cont.)

VARIABLES USED

= 11 COTH; 12 ASINH
 = 13 ACOSH; 14 ATANH
 = 15 ACOTH; 16 DEVNOR
 = 17 ABSOLUTE; 18 EXPONENT
 = 19 LOGE; 20 SIN
 = 21 COS; 22 TAN
 = 23 COT; 24 ARCSIN
 = 25 ARCCOS; 26 ARCTAN
 = 27 ARCCOT; 28 SIND
 = 29 COSD; 30 TAND
 = 31 COTD; 32 ASIND
 = 33 ACOSD; 34 ATAND
 = 35 ACOTD; 36 ASIN
 = 37 ACOS; 38 ATAN
 = 39 ACOT

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
21		Check number of arguments.
24	10	IL will contain the address of the storage column.
30	40	ILZ will mark the end of the storage column.
31		The value of NARGS will be one less than the number of arguments.

FUNCT (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
32-36	45,50	This loop obtains, for each remaining argument, the address of either a column or a constant depending on the type of argument.
37-39		If a trigonometric function is followed by the qualifier DEGREE, the value of L2, which is used to determine which evaluation is to be performed, must be modified.
41,42		If the first argument is real then the function needs to be evaluated just once. LOCRTN equal to 1 indicates this situation. Else a whole column of arguments is used.
43-45		The value of L2 determines which function is to be evaluated.
46	52	ARGS(1) will now contain the value of the function at the desired point when the first argument is a constant.
52		This ASSIGN statement is used when the first argument is a column number and will initiate the function evaluation for each value in the column.

FUNCT (cont.)

58,59

When there are two arguments and the first is a constant, store the functional value in the designated column.

63

70

When there are two arguments and the first is a column number, LOCRTN is 2.

64

I will be used as a subscript to obtain from the worksheet the input to be evaluated.

65

80

Check to see if the end of the column has been reached.

66

X will be the argument in the function evaluation.

67-69

GO TO the section of the program in which the desired evaluation will be performed.

70-73

The program returns to this place after the function has been evaluated. Result is placed into the worksheet and subscripts are incremented.

74

K2 will be used to increment the subscript for the second argument in the three-argument case and thus must be zero if the second argument is a constant.

79-82

100

Complete the required computations for the three-argument case where the first

FUNCT (cont.)

argument is a constant.

86-97 110,120,115 This section is used for the three-argument case when the first argument is a column number.

99-102 250 This statement is executed only when evaluations must be made for a column, and passes control to the appropriate section where the label for the beginning of the section will be assigned to INDEX for use in the assigned GO TO in line 67 or 90.

104 275 Pass on to the next row of the worksheet, if necessary.

105-273 299-610 The calculations for each function are performed in this section of the program.

GENER

PURPOSE: This subroutine is used for executing the command GENERATE

COMMON BLOCK

BLOCKA

BLOCKD

VARIABLES USED

NFLAG

RC, IARGS, KIND, NRMAY, NROW, NARGS

GENER (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
9		Check for illegal number of arguments.
13	20	Obtain address of column.
19-21	30,40	All arguments must be floating point or converted to floating point numbers.
22-27	50	Check that the end points and increments are legal and find out how many rows would be needed.
28-29		If the GENERATE command requires more rows than are available, the user is asked to cancel the command or have it executed as far as possible.
32-45	110-150	Values are actually generated into the specified column.

INPUT

PURPOSE: This subroutine reads a line from the CRT reply area.
The character string is stored in NEWCD and is converted into
a numerical code stored in KARD.

COMMON BLOCK

BLOCKA

VARIABLES USED

KARD, NEWCD, KRDEND

INVCHK(NB,DET,JP)

PURPOSE: This subroutine is used for moving a matrix into a scratch area prior to inverting it. An identity matrix is also placed into the scratch area since Gaussian elimination is used as the inversion method.

COMMON BLOCK

BLOCKD

BLOCKE

SCRAT

VARIABLES USED

RC, IARGS, NROW

L2

A (equivalenced to B)

LINE
NUMBERFORTRAN
LABELCOMMENTS

20

NA will be the dimension of the matrix.

22

JC will point to the right hand side
vector.

23,24

Initialize some constants.

25

JAP will be the address of the matrix.

26-40

9-12

The DO 10 loop sets up NA records on a
scratch data set. Each record contains
a row of the matrix to be inverted;
this row is obtained in the DO 9 loop
and a row of the identity matrix is
generated in the DO 12 loop. For
solving a system of linear equations,
the right hand side vector is attached
to the matrix.

INVCHK (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
42-46	13	Create the last row for the commands LINEAR and MLINEAR.
47	14	Call SPINV to invert the matrix.

INVERT

PURPOSE: This subroutine is called in response to the commands MINVERT, INVERT, MLINEAR and LINEAR and checks arguments and stores the results. (The calculations take place in the SPINV subroutine.)

COMMON BLOCK

BLOCKA

BLOCKD

SCRAT

BLOCKE

VARIABLES USED

NFLAG

RC, IARGS, KIND, NARGS, NROW

A (equivalenced to B)

L2 = 1 INVERT, MINVERT

= 2 LINEAR, MLINEAR

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
15		Check the number of arguments.
18-20	1200	Check for illegal arguments.
22-25		Expand the five - argument form into an equivalent six - argument form.

INVERT (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
21-30	90	There is one matrix to check for LINEAR and MLINEAR and two matrices to check for INVERT and MINVERT. The implied dimensions of the second matrix are placed into the argument array.
31,32	95	Check the legality of the matrices.
33	96	A 15 by 15 matrix is the largest matrix inverted by this system.
34-39		M1 will contain the dimension of the matrix to be inverted. For solving a system of linear equations the dimen- sion is one larger than the dimension of the matrix of coefficients and one needs to obtain column addresses for the last two arguments.
44		Non-zero determinant implies successful inversion.
45,46		Set constants which will be used in storing results.
49-57	100,110	The inverse of the matrix is placed into the worksheet.
60-63	130,140	Store the solution of a system of linear equations.

INVERT (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
64-69	150,160	Write out the determinant of the matrix on the screen.

LOOKUP

PURPOSE: This subroutine contains the dictionary of commands. An entered command is compared with the entries in the dictionary and indicators are set to identify the command.

COMMON BLOCK

BLOCKE

VARIABLES USED

NAME, L1, L2

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
13-98		These data statements create a dictionary of commands.
106-239	104-360	Compare the command given by the user with the dictionary of commands. Set L1 and L2 to identify the command.
240	899	Set L1 to 0 if the command is not in the dictionary.

MAIN

PURPOSE: This program contains a cross-reference table showing for each labelled COMMON which subprogram it is used in. It also shows for each subprogram those subprograms which reference it. It prepares a random access file and a plotting file, initializes GMS, calls the OMNITAB driver routine, releases GMS and closes the plotting file.

COMMON BLOCK

BLANK

VARIABLES USED

IOVLY

MATRIX

PURPOSE: This subroutine is used in executing the commands MADD, MSUB, MTRANS, ATRANS, AADD, ASUB, AMULT, ADIVIDE, ARAISE, ASCALAR, MSCALAR and SCALAR.

COMMON BLOCK

BLOCKA

BLOCKD

SCRAT

BLOCKE

VARIABLES USED

NFLAG

RC, IARGS, KIND, NARGS, NROW

A

L2 = 1 MADD

= 3 MTRANS, ATRANS

= 4 AADD

= 5 ASUB

= 6 AMULT

= 7 ADIVIDE

MATRIX (cont.)

COMMON BLOCKVARIABLES USED

= 8 ARAISE

= 9 ASCALAR, MSCALAR, SCALAR

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
22-24		Initialize some constants.
30-35	100-140	Check the number of arguments for the various commands.
42-44	605-610	All arguments at this point should appear to be integer. This check will also indicate an error if an excessive number of arguments are used for ASCALAR, MSCALAR and SCALAR.
47	640	N2 points at either a constant or a column for the cases in which not all arguments define matrices.
48		For ASCALAR, MSCALAR and SCALAR the N2 argument must be a constant.
53-61	660,680	Check the N2 argument. IBP will point to either a constant or a column. Manipu- late the argument arrays so as to avoid improper error detection later.
66-69	850	This refers to the situation where only one dimension is given for the matrix. Thus all arguments except the first two

MATRIX (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
		must be moved out one place in the IARGS array. Thus IARGS(4) will equal IARGS(3).
71-79	1300	The dimensions of the second matrix are not given. Arguments 7 and 8 must be moved into 9 and 10 so that the implied dimensions can be stored in 7 and 8. For MTRANS and ATRANS the number of rows of the first becomes the number of columns of the second and vice versa. Also NROWPP, a stepping constant, must be changed from NROW to 1.
88-89	1600	The dimensions of a third matrix must be picked up from the dimensions of the first.
91-92	1700	Check whether the matrices fit in the worksheet. J, the number of matrices, is either 2 or 3.
97-105	1400,2000	Initialize some variables which will be used in performing the required calculations.
106-138	2100-3560	The required calculations are performed and the results are stored on a scratch

MATRIX (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
		data set.
142-150	4060,4080	The results are moved from the scratch data set into the worksheet.

MDAMAD

PURPOSE: This subroutine is used for executing the commands M(AD) and M(DA).

<u>COMMON BLOCK</u>	<u>VARIABLES USED</u>
BLOCKA	NFLAG
BLOCKD	RC, IARGS, NROW, NARGS
BLOCKE	L2 = 4 M(AD) = 5 M(DA)
SCRATCH	A

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
16		There must be exactly seven arguments.
20-23		All arguments must be integers.
27		YDP will contain the address of the column.
31-34	50	The coordinates of the result must be in IARGS(5) and IARGS(6). Pick up the dimensions from IARGS(3) and IARGS(4).
35-37		Both matrices must fit into the worksheet.

MDAMAD (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
46-55	200,260	Initialize variables used in the calculations. I1 and I2 are used to increment subscripts and must be set to 1,0 for AD and to 0,1 for DA.
56-65	300,400	The required calculations are performed and the results are placed in a scratch data set.
66-73	400,440	The results are placed into the specified part of the worksheet.

MISC2

PURPOSE: This subroutine is used for executing the commands CLOSE, COUNT, SHORTEN, EXPAND and DUPLICATE.

COMMON BLOCK

BLOCKA

BLOCKD

SCRAT

BLOCKE

VARIABLES USED

NFLAG

RC, IARGS, KIND, NRMAX, NARGS, NROW, NCOL

A

L2 = 1 CLOSE

= 2 COUNT

= 3 SHORTEN

= 4 EXPAND

= 5 DUPLICATE

MISC2 (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
20		There must be at least two arguments.
25-28	50-70	L2 argument must be a constant (floating point) for CLOSE(L2=1) and SHORTEN (L2=3). KIND(L2) is set to 0 for use in subsequent checks.
29		SHORTEN requires exactly five arguments.
30		Store the one floating point argument in ARG1.
33	74	COUNT must have exactly two arguments.
34-35		All arguments for CLOSE, COUNT and SHORTEN should appear to be legitimate column numbers.
36-37	90	Adjust elements of IARGS so that they may be used most easily in executing the command.
47-48	140	The command must be applied to columns IARGS(2) through IARGS(NARGS). K will point to the column being acted upon at the moment.
52	148	Compare each value in a column with the test value.
54		Get out of loop if all rows in a column have been checked.

MISC2 (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
55-58	155	J1 points at the value to be deleted. Each value below is moved up one row.
62-65	180	Fill out the column with zeros.
71-79	200-260	Execute the command COUNT by searching for a non-zero value starting from the bottom of the specified column.
83-100	300-380	Execute the command SHORTEN.
84-93	320-360	Search for the desired truncation point, ARG1. Reset NRMAX accordingly.
94-100	370,380	Place the shortened columns into the designated columns.
105	400	The command EXPAND requires exactly four arguments.
106,107		The second and third arguments are expect- ed to be floating point numbers.
108		Check to see if there are enough columns for the results.
109-111		K1 plus the index of a DO statement will be used to insert results in the specified columns of the worksheet.
113-118	450	If the first argument is a column number, transfer the values into the scratch array A.

MISC2 (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
120,121	460,470	If the first argument is a constant, transfer the values into A.
130-138	570,580	Complete execution of the command EXPAND.
143-145	600	The command DUPLICATE must have exactly seven integer arguments.
146		Let the ninth argument be the number of duplications to be performed.
147-148	630	Arguments 2 through 7 become arguments 1 through 6.
149-153		Arguments 7 and 8 are implied and must be supplied for checking.
156		The number of duplications must be at least one.
160-162		Set constants to be used in executing the command.
163-169		Place the array to be duplicated onto a scratch data set.
170-180		Execute the command.

MULT

PURPOSE: This subroutine is used for the execution of the command
MULT.

MMULT (cont.)

COMMON BLOCK

BLOCKA

BLOCKD

SCRAT

VARIABLES USED

NFLAG

RC, IARGS, NROW, NARGS

A

LINE
NUMBERFORTTRAN
LABELCOMMENTS

10

IROWA is the number of rows of the resulting matrix.

14

Check the number of arguments.

18-20

600

All arguments must be integers.

25-36

820-831

If there are fewer than ten arguments, then manipulate the argument list so as to simulate the equivalent ten-argument form.

38

840

In the ten-argument form, the fourth and seventh arguments must agree.

39

1100

ICOLB is the number of columns of the resulting matrix.

40

15 rows or 15 columns is the limit on the size of the product matrix.

41-42

Arguments 11 and 12 must contain the dimensions of the resulting matrix.

43-45

Check that all three matrices fit within the worksheet.

51-66

3000-3040

Perform the matrix multiplication and

MMULT (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
		store the results in a scratch data set.
70-78	8080,8100	The results are placed into the designated location in the worksheet.

MOP

PURPOSE: This subroutine is used for the execution of the commands MDEFINE, ADEFINE, MZERO, AZERO, MERASE, AERASE, MIDENT, MDIAG, ADIAG and MTRACE.

<u>COMMON BLOCK</u>	<u>VARIABLES USED</u>
BLOCKA	NFLAG
BLOCKD	RC, IARGS, KIND, NROW, NARGS
SCRAT	A
BLOCKE	L2 = 1 MDEFINE = 2 ADEFINE = 3 ADIAG = 4 MDIAG = 5 MZERO = 6 AZERO = 7 MERASE = 8 AERASE = 9 MIDENT = 10 MTRACE

MOP (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
21	100	For the commands MDEFINE and ADEFINE, there must be either four or five arguments.
22		The last argument must be a floating point number.
23		In the four-argument form, set the fourth integer argument to the value of the third integer argument. The fourth entered argument has to be floating point and thus is stored in the array ARGS.
24,25		Set constants for later use.
26		The first J arguments must be integers. For this command, the last argument will not be an integer.
27-32	105	This code is used, for all the commands executed in this subroutine, to check that the required arguments are integers and that the matrix fits into the work- sheet.
35		JB is the beginning of the matrix in the worksheet.
37		N is the number of rows in the matrix.

MOP (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
36		Nothing more is necessary to execute the command MTRACE (L2=10).
40-48	110,120	CONST is placed into the KA'th column of the matrix and then CONSTA is placed into the KA'th row of the column if required.
49		Additional code must be executed for the commands MDIAG and ADIAG (L2=3,4).
51-57	150	For the commands MZERO, MERASE, AZERO and AERASE, check for errors and set constants to 0.
58-69	160-170	For the command MIDENT, set constants to 0 and 1 and check arguments.
66-79	180-188	For the commands MDIAG and ADIAG, set constants to 0 and, if the last argu- ment is a column number, store the column in the scratch array A. Also check arguments for errors.
82-85	200	Place the designated constant into the diagonal of the matrix for the commands MDIAG and ADIAG.
87-89	220,230	Place the designated column into the diagonal of the specified matrix.

MOP (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
91-98	250	Check arguments and prepare for execution of the command MTRACE.
100-104	260,270	Calculate the trace of a matrix.

MOVE

PURPOSE: This subroutine is used for executing the commands MOVE,
BLOCKTRANSFER, AMOVE and MMOVE.

COMMON BLOCK

BLOCKA

BLOCKD

SCRAT

VARIABLES USED

NFLAG

RC, IARGS, NROW, NARGS

A

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>
------------------------	--------------------------

COMMENTS

13

There must be exactly six arguments.

23-25

70

All arguments must be integers.

26-27

The dimensions of the second matrix are
taken from those of the first.

28-30

Check that the two matrices fit into the
worksheet.

33-40

100,110

Copy the first matrix onto a scratch data
set.

MOVE (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
41-48	200,210	Replace the second matrix by the one in the scratch data set.

MRAISE .

PURPOSE: This subroutine is used for executing the command MRAISE.

COMMON BLOCK

BLOCKA

BLOCKD

SCRAT

VARIABLES USED

NFLAG

RC, IARGS, KIND, NROW, NARGS

A

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
12		ISIZE will contain the dimension of the matrices which must be square.
16		There must be either six or seven arguments
20		J points to the power to which the matrix is to be raised.
22,23		The ninth argument will now contain the power which must be at least one.
24-26		All arguments must be integers.
29-31		If the power is a floating point number, change it to an integer so that the above checks will not detect that it was entered as a floating point number.

MRAISE (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
36		Check for squareness in the seven-argument form.
37,38		In the seven-argument form, the sixth and seventh arguments become the fifth and sixth arguments.
40-42	1100,1150	Set the required arguments equal to the given dimension of the matrix.
43-45		Check to see if the two matrices fit into the worksheet.
51		NPOW will be the number of matrix multiplications to be performed.
53		If the matrix is to be raised to the first power and the two specified matrices are the same, nothing needs to be done.
54-64	4030,4040	Move the matrix to the specified location since no multiplication is required.
66-95	4050-5040	This loop forces the required multiplication to be done the specified number of times.
67		ISAVP will point toward the result matrix.
68-71	4060	IRP points toward the matrix which resulted from the previous matrix multiplication and must initially point at the matrix

MRAISE (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
		specified first in the argument list.
72-95	4070-5040	Perform the matrix multiplication.
79-82	4080	The row from the previous step must be saved in a scratch array since the row will be replaced as each element is obtained.

MSCROW

PURPOSE: This subroutine is used for executing the commands PARSUM,
PARPROD, RMS, AVERAGE and SUM.

COMMON BLOCK

BLOCKA

BLOCKD

BLOCKE

VARIABLES USED

NFLAG

RC, IARGS, KIND, NRMAX, NARGS, NROW

L2 = 1 PARSUM

= 2 PARPROD

= 3 RMS

= 4 AVERAGE

= 5 SUM

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
12		ELEM will be used for summing a column.
16	40	Obtain the address of the first column

MSCROW (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
		(J1).
20	60	Obtain the address of the result column (J2).
25,26	140	Three or more arguments are legal only for the command SUM (L2 = 5).
28-31	100	All arguments between the first and last must be row numbers.
32		For the four-argument form of SUM, the second argument cannot exceed the third.
38-42	155	For the four-argument form, sum from row I2 to row I3.
45	160	Store the result in the indicated column of the worksheet.
48-50	170,190	Sum the values in the indicated rows.
60-70	220-240	Obtain partial sums and partial products.
75-78	280,290	Obtain RMS.
83-85	300,310	Sum over all rows of a column.
87		Obtain a column average.

NTXCHK (J)

PURPOSE: This subroutine checks to see if the first J matrices defined in the argument list fit within the worksheet and locates the starting point of each one within the worksheet

MTXCHK (J) (cont.)

array RC.

COMMON BLOCK

BLOCKD

VARIABLES USED

IARGS, KIND, NROW, NCOL

LINE
NUMBERFORTTRAN
LABELCOMMENTS

20

J is the number of matrices to be checked,
thus JB is the number of arguments
required.

21

J will be 0 if no error is detected.

22-26

100

J is set to 1 if a negative argument is
encountered.

27-32

120

Check that each matrix fits into the
worksheet and set IARGS(I) to point
to the upper left hand corner of the
matrix.

34

130

J is set to 2 if a matrix overflows the
worksheet.

MTX

PURPOSE: This subroutine is used for executing the commands N(XX')
and N(X'X).

COMMON BLOCK

BLOCKA

VARIABLES USED

NFLAG

COMMON BLOCK

BLOCKD

BLOCKF

SCRAT

BLOCKE

MXTX (cont.)

VARIABLES USED

RC, IARGS, NARGS, NROW

NCTOP

A

L2 = 1 M(XX')

= 2 M(X'X)

Note: For other values of L2, subroutines are called which execute the commands M(X'AX), M(XAX'), M(AD), M(AV), M(V'A).

LINE
NUMBERFORTTRAN
LABELCOMMENTS

19-21

10,20

When L2 is 2, the command was M X. If the number of arguments is six or less, then the command is assumed to be M(X'X). Otherwise, the command is assumed to be M(X'AX).

23

40

Call the subroutine MDAMAD for the commands M(AD) and M(DA).

25

60

Call the subroutine ARYVEC for the commands M(AV) or M(V'A).

27

100

The commands M(XX') and M(X'X) must have either five or six arguments.

31-33

All arguments must be integers.

MXTX (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
38-41		Transform the five-argument form into an equivalent six-argument form.
42-44		Obtain the implied dimensions of the resulting matrix and check to see that it is not too large.
47-49	200	Check to see that the two matrices fit within the worksheet.
59-62		Prepare constants for the command M(XX').
64-67	320	Prepare constants for the command M(X'X).
68-85	340-440	Perform the matrix multiplication and store the results in the scratch data set.
89-99	500,520	Place the results into the designated location in the worksheet.

NNAME (NAME)

PURPOSE: This subroutine converts a string of up to six letters into
two numerical values with the first three letters determining
NAME(1) and the last three determining NAME(2).

COMMON BLOCK

BLOCKA

VARIABLES USED

N, KARD

NNAME (NAME) (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
41,42	10	Elements of MISC not changed later must be 0.
43-47	20	The array KARD contains a numerical representation of the input line. Up to six characters will be checked. Translation stops if a non-letter is found. The value of MISC(I) is the position of the i'th letter of the string in the alphabet, i.e. MISC(I)=1 for A and MISC(I)=26 for Z.
48-50	30	Scan for the first non-letter following the string.
51,52		The NAME array contains two values which together uniquely identify the letter string.

NONBLA(I)

PURPOSE: This function subprogram searches for a non-blank character starting with the M'th. The value returned will identify the character and M will indicate its position.

COMMON BLOCK

BLOCKA

VARIABLES USED

N, KARD

OMCONV (NWCD, KRD, KRDEND)

PURPOSE: This subroutine takes an array of KRDEND characters in NWCD and converts them into a numerical code in KRD.

<u>LINE NUMBER</u>	<u>COMMENTS</u>
4,5	Store the addresses of NWCD and KRD in registers 3 and 4.
6	Move 80 bytes from NWCD to KRD.
8-10	Store constants 0, 1, 4 in registers 5, 7, 8.
11	Store the address of KRDEND in register 9.
12	Load the value of KRDEND into register 11.
13	Register 6 will be used to control the BCT instruction below.
14	The logs will be executed KRDEND-1 times.
15,16	The first KRDEND-1 bytes obtained in the translation must be moved so that there are three bytes between each of them. The move must be performed from right to left and register 11 will contain the relative address of the storage byte.
17-20	This loop actually moves the required bytes using register 5 to zero out the three bytes between successive non-zero bytes.
21,22	Performs a last move.

OMCONV (NWCD, KRD, KRDEND) (cont.)

LINE
NUMBERCOMMENTS

24-42

Set up the translation tables.

OMNIT

PURPOSE: This is the principal subroutine and controls execution of the entire program.

COMMON BLOCKVARIABLES USED

BLOCKA

MODE, M, KARD, KARG, ARG, ARG2, NEWCD,
NEWCD5, KSAVE, NSAVE, NFLAG

BLOCKD

ARGTAB, NARGS

BLOCKE

NAME, L1, ISRFLG

KPLOT

NFRAME, KKND, SIZE, SPACE

QRS

JROW

BLANK

KEY, IOVLY, ITYPE

LINE
NUMBERFORTRAN
LABELCOMMENTS

25

4 is the unit number for a data set used
in displaying desired text on the CRT
screen.

26-28

Initialize constants.

29-30

Prepare for interrupts.

31

Present the initial display of instructions.

OMNIT (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
32-35	50	The NAME array must be zeroed out before a new name is read.
36		Start the number of arguments, NARGS, at 0.
37		J will be the index of an array ARG TAB which contains information about the arguments of a command.
38	52	If KEY is 31 then return to the GMS monitor.
42		Write READY on the screen except when the previous command was not executed or was MINVERT, INVERT, LINEAR or MLINEAR or when in input mode activated by READ command.
50-56	524,5240,999	Write out, on the screen, the row number of the next row to be entered.
61	525	Await user's command.
63-74	53,535	These statements process interrupts from the programmed function keyboard.
75	54	Call the subroutine INPUT to process a command entered from the regular key- board.
82-84	55	M will be incremented so as to enable the entire line to be read.

OMNIT (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
85		Ship all special characters except for \$. If \$ is encountered then processing for the current line may cease.
87,88		Numbers in the line before the command are illegal except in the input mode.
92	70	When a letter is found, call the sub- routine NNAME to compile it and store its numerical equivalent in NAME(1) and NAME(2).
100-102		When the command is OMNITAB, reset certain variables and restart.
106,107	87	When the command is STOP, return control to the GMS monitor.
111-119	88,884,885	The command ROW is legal only when in the READ initiated input mode. Determine its argument and reset the row counter, JROW, accordingly.
129		Branch to 100 for numbers and to 90 for letters.
130		Branch to 100 for asterisks.
131		Branch to 200 for end of line.

OMNIT (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
137	90	A second name following the command may be a command qualifier and must be treated as the command was treated.
142-143		At least one character can be skipped if the command was M.
157		Call the subroutine AARGS when a string of numbers is found.
159,160	103	When a floating point number is found, set the J'th element of ARGTAB to 0. The actual number will be the next element in ARGTAB.
166,167	105	Add 8192 to an integer argument and check that it is greater than 0. This will distinguish it later from other types of arguments.
170-172	110,115	Place the assembled argument into ARGTAB and increase by 1 the number of arguments.
180-185	120,125	KARG=1 if only one asterisk is found. KARG=0 if two asterisks are found. The subroutine ASTER is used to assemble arguments involving asterisks.

OMNIT (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
205	135	The value of ARGTAB(J) will indicate the variable and its type.
207-211	140	A worksheet reference requires a pair of values in ARGTAB. The sign of the second indicates whether the worksheet reference is to be floating point or integer.
213-216	150,155	If a string of three or more asterisks is found, set ARGTAB(J) to -1, except when J is 1. An error occurs when an asterisk string is not preceded by an argument.
265	202	Call the subroutine EXPAND to convert the information in ARGTAB into a form which is used by the subroutines which execute the commands.
266-269	204	Data enters the worksheet following a READ(ISRFLG=0) or SET(ISRFLG=1) command.
271-282	9002-9006	The entered line is saved for later recall.
287-289	210	Check name against dictionary of names by calling LOOKUP. If LI is not 0, the name was found in the dictionary. If

OMNIT (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
		no name is found, an error results if the program is not in the input mode.
295	220	Reset MODE to 1, the interpretive mode.
296	222	See comment for line 265.
297		Call the subroutine XECUTE which calls the appropriate subroutine necessary to execute the given command.
298		Go back to the beginning for the next command.

PDMOTE

PURPOSE: This subroutine is used for executing the commands PROMOTE
and DENOTE.

COMMON BLOCK

BLOCKA

BLOCKD

BLOCKE

VARIABLES USED

NPLAG

RC, IARGS, NIMAX, NROW, NARGS, NCOL

L2 = 10 PROMOTE

= 11 DENOTE

PDMOTE (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
13		L2 is transformed to 0 for PROMOTE and to 1 for DEMOTE.
14		There must be an odd number of arguments.
18	30	NR is the length of the shift.
20-22	31	Shift all the arguments but the first. Thus the first NARGS(NARGS-1) arguments should be column numbers.
26,27	32	Check to see if the arguments are logiti- mate column numbers.
34-36	40	If the shift is negative, change it to be positive. The value of L2 must also be changed.
41		For the command DEMOTE, check that the execution will not reach beyond the end of any column.
48-52	95	If the only argument for PROMOTE is NRMAX, then the entire worksheet will be zeroed out.
54,55	100	LIMIT is twice the number of columns to be promoted or demoted. If no columns are specified, then all columns are to be used.

PDMOTE (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
64-66		Set constants when no columns are specified.
68,69	120	Set constants when columns are specified.
74-79	140	Execute the command DEMOTE.
84-89	150,160	In response to the command PROMOTE, move the first column into the second of a pair of columns. The first NR will be lost.
94-97	170	If columns are specified, fill the bottom of the second of a pair of columns with 0's.
99		If the command was DEMOTE, then NRMAX must be increased.

PFINT

PURPOSE: This subroutine sets ITYPE to 1 when the wait state is interrupted by depression of a programmed function key. KEY is set to the number of the key which was depressed.

COMMON BLOCK

BLANK

VARIABLES USED

ITYPE, KEY

PHYCON (NAME)

PURPOSE: This subroutine locates a physical constant when one is used and finds its value.

COMMON BLOCK

BLOCKA

PCONST

VARIABLES USED

ARG

P,N

LINE
NUMBERFORTTRAN
LABELCOMMENTS

9-12

20

Check to see if NAME is a predefined
constant.

13

Set ARG to 0 if NAME does not correspond
to a name in the constant table N.

15

Obtain the value of the constant from P.

PLBK

PURPOSE: This subroutine displays a command after it has been entered and gives the user a chance to check it before having it executed.

COMMON BLOCK

BLOCKA

KPLLOT

BLANK

VARIABLES USED

NEWCD, NFLAG

NFRAME, KKND, SIZE, SPACE

KEY, ITYPE

PLBK (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
11		Display the line which the user has just entered.
13-16		Inform the user that there is no syntax error and ask him to confirm the command or cancel it.
23-24		Transmit the CRT image to a data set for plotting.
27	1448	NFLAG is set to 1 to halt execution of the command.
28		Erase the command from the screen.

PRGRAM(I01)

PURPOSE: This subroutine is used for displaying the program which the user has written.

<u>COMMON BLOCK</u>	<u>VARIABLES USED</u>
BLOCKA	NEWCD5, KSAVE, NSAVE
KPLOT	NFRAME, KKND, SIZE, SPACE
BLANK	KEY, IOVLY, ITYPE

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
13-22	2,25	The user is given a chance to see the commands he has entered if he fills

PRGRAM(I01) (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
		up the space for storing commands. There is space available for 200 lines.
23-25 47-49 66-68		Transmit CRT image to a data set for plotting on CALCOMP.
11		I01 will be 0 when the user decides to list his program on the screen.
34		IOVLY is the associated variable.
35-41	40	Read the entered program from data set NSAVE and write it onto the screen.
55-60		Write out any lines which have not yet been stored on the data set.

PROROW

PURPOSE: This subroutine is used for executing the commands
ROWSUM and PRODUCT.

COMMON BLOCK

BLOCKA

BLOCKD

BLOCKE

VARIABLES USED

NFLAG

RC, IARGS, NRMAX, NROW, NARGS

L2 = 1 ROWSUM

= 2 PRODUCT

PROROW (cont.)

COMMON BLOCKVARIABLES USED

SCRAT

A

LINE
NUMBERFORTTRAN
LABELCOMMENTS

13-22

40-60

Check for errors.

25,26

CONST must be 0 for ROWSUM and 1 for
PRODUCT.

27,28

Rowsums and products are accumulated in
the array A.

30-37

140,150

Obtain row sums or row products for the
three-argument form where IA1 is the
beginning of the first column and IA2
is the beginning of the second column
in the argument list. Accumulate the
results in the scratch array A.

38-41

170,180

Store the results into specified column
in the worksheet.

43-52

200-250

Obtain row sums or row products when
specific columns rather than a range
of columns is specified.

READQ

PURPOSE: This subroutine is used when the program is in input mode to set a line of data into the appropriate row following a READ command.

COMMON BLOCK

BLOCKA

BLOCKD

QRS

VARIABLES USED

NEWCD

RC, IARGS, KIND, NRMAX, NROW

J, NNARG

LINE
NUMBERFORTRAN
LABELCOMMENTS

19-30

A row of data is entered into the worksheet.

27

Write the row of data onto the screen.

31,32

J contains the number of rows entered.
NRMAX is adjusted, if necessary.

READX

PURPOSE: This subroutine is called to execute the command READ.

COMMON BLOCK

BLOCKA

BLOCKD

VARIABLES USED

MODE, NEWCD

ARGS (equivalenced to the end of RC),

IARGS, NARGS

READX (cont.)

COMMON BLOCKVARIABLES USED

BLOCKE

ISRFLG

QRS

J, NNARG

LINE
NUMBERFORTTRAN
LABELCOMMENTS

13-19

5-15

Check for errors.

24

MODE = 2 indicates that data are entered.

32-35

30

Process the argument list.

33

Column addresses are stored starting at
the 40th element of IARGS.

37

NNARG will contain the number of columns
into which data are entered.

RESET

PURPOSE: This subroutine is used for executing the command RESET.COMMON BLOCKVARIABLES USED

BLOCKA

NFLAG

BLOCKD

IARGS, ARGS (equivalenced to the end of
RG), KIND, NIMAX, NROW, NARGS, VWXYZ

BLOCKE

L2 = 1 V

= 2 W

= 3 X

= 4 Y

RESET (cont.)

COMMON BLOCKVARIABLES USED

= 5 Z

= 6 NRMAX

LINE
NUMBERFORTTRAN
LABELCOMMENTS

12

Only one argument is allowed.

19

30

NRMAX is an integer. Thus a real argument must be transformed into an integer argument.

25

Reset NRMAX.

32

Real arguments are expected. Transform integer arguments into real ones.

33

Reset the designated variable.

SCRAM(NC,IT)

PURPOSE: This subroutine is used to read and write from a scratch file when the scratch array A alone is not sufficiently large.

COMMON BLOCKVARIABLES USED

BLOCKA

NSAVE

BLANK

IOVLY

SCRAT

A

SCRAM(NC,IT) (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
4		IPTR is used to hold the value of the associated variable IOVLY which is used for storing the lines of a program.
5		The first forty records are reserved for entered program lines.
6		IT is 2 for write and 1 for read.

SET

PURPOSE: This subroutine is called in response to the command
SET.

COMMON BLOCK

BLOCKA

BLOCKD

BLOCKE

QRS

VARIABLES USED

NFLAG, MODE

IARGS, KIND, NROW, NARGS

ISRFLG

NDROW, J

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
20	20	NDROW marks the end location of the column in the worksheet.

SET (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
26	24	J, which marks the location at which storage is to begin, must be increas- ed if row 1 is not the first row to be used.
29		ISRFLG=1 indicates SET command.
30		MODE=2 is the data input mode.

SETQ

PURPOSE: This subroutine is used for entering data into the
worksheet following a SET command.

<u>COMMON BLOCK</u>	<u>VARIABLES USED</u>
BLOCKA	NFLAG
BLOCKD	IARGS, KIND, NRMAL, NROW, NARGS
QRS	NDROW, J

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
17		J and JJ mark the first and last rows into which the arguments will be placed.

SETQ (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
18-20		If there are too many arguments to fit into the column, the user is notified of this fact and may cancel the command.
24,25		If some of the arguments can be inserted into the column, JJ is reset to the end of the column.
28-35	15,20,30	The arguments are entered into the worksheet.
36		J is reset to mark the row where storage will resume with the next command.
37		Reset NRMAX if its current value has been exceeded.

SORDER

PURPOSE: This subroutine is used for executing the commands SORT, ORDER, HIERARCHY.

COMMON BLOCK

BLOCKA

BLOCKD

SCRAT

VARIABLES USED

NPLAG

RC, IARGS, NRMAX, NARGS

A

SORDER (cont.)

COMMON BLOCK

BLOCKE

VARIABLES USED

L2 = 8 SORT

= 9 ORDER

=14 HIERARCHY

LINE
NUMBERFORTRAN
LABELCOMMENTS

14-21

10-50

Check for errors.

22,23

60

The command HIERARCHY (L2=14) must have
exactly two arguments.

29-32

If NRMAX=1, do nothing for SORT and ORDER
or place a 1 in the column indicated
by the second argument for HIERARCHY.

35-38

130,140

Place the column into A and the row
numbers into NUM.

39-53

160,200

Order the columns from low to high in A.
NUM(1) will contain the row number
of the I'th ordered value.

54-59

210,230

Completed the HIERARCHY command by place-
ing the ranks of the first column into
the second column.

60-62

240,250

Replace the original column by the ordered
column.

63

Nothing else needs to be done if there is
just one argument.

SORDER (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
64-68		Prepare to sort the next column.
69-77	290-310	Rearrange subsequent columns using the ordering obtained for the first.

SPINV(M,DET)

PURPOSE: This subroutine inverts a matrix of dimension M using
Gaussian elimination and calculates its determinant, DET.

COMMON BLOCK

SCRAT

VARIABLES USED

A (equivalenced with B)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
12		Initialize DET to 1 so that the value of the determinant can be determined later by multiplying the pivotal elements.
13		N is the number of rows.
14		N2 is the number of columns.
15-36	12,13,20	Search for the largest element in the L'th column. Start initially at column 1.
41,42	30	C is the largest element in the C'th column. If it is 0, then the matrix is singular and the determinant is 0.

SPINV(M,DET) (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
48	22	Check to see if any interchanging of rows is necessary. The L'th row must contain the largest element in the L'th column.
49-56	24,25	Switch row J1 and row L.
60-80	32,3235,321- 325	Zero out all elements except the pivotal element in the Lth column.
76,77		Check for near singularity.
85-92		Divide by the pivotal elements and calculate the determinant.

STATD

PURPOSE: This subroutine is used for executing the commands YORMX,
YORMP, YORMZ, GAMX, GAMP, GANZ, CHIX, CHIP, CHIZ, TTX, TTP,
TTZ, BETAX, BETAP, BETAZ, PFX, PPP, PFZ.

<u>COMMON BLOCK</u>	<u>VARIABLES USED</u>
BLOCKA	NFLAG
BLOCKD	RC, NRMAL, NARGS
BLOCKE	L2 = 1 YORMX = 2 YORMP = 3 YORMZ = 4 GAMX = 5 GAMP = 6 GANZ

STATD (cont.)

COMMON BLOCKVARIABLES USED

= 7 CHIX	= 8 CHIP
= 9 CHIZ	=10 TTX
=11 TTP	=12 TTZ
=13 BETAX	=14 BETAP
=15 BETAZ	=16 FFX
=17 FFP	=18 FFZ

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
15-17		Check for correct number of arguments.
18,19		IL will point to the beginning of the column in which the results are to be placed.
24	40	ILZ will point to the end of the column of results.
26-32	45,50	Convert all arguments except for the last one into a form which can be used by the execution loops later. Both constants and column numbers are acceptable arguments.
40,41		J will be 1 if all arguments but the last are constants. The function will only be evaluated once if INDEX1 is 100 and repeatedly if INDEX1 is 105.

STATD (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
42,43		This computed GO TO is used only once and passes control to the section of the subroutine in which the required evaluation is performed.
44-49	80-90	Obtain the next arguments to supply to the subroutines which execute the required task.
51	100	Y is the value of the function with the given arguments. Place it in the designated column.
53-55		Place the value of the function into a row of the designated column and increment IL so that the next row will be used next. If the end of the column has been exceeded then execution is finished.
56,57		This assigned GO TO is used for all arguments but the first and passes control to the required section of the worksheet.
58-60	903	Y is set to 0 if an arithmetic error occurs and execution continues.

STATD (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
61-130	110-281	Each of the commands is executed by a call to a subroutine within this section of the program.

TRANSF

PURPOSE: This subroutine is used for executing the commands
M(X'AX) and M(XAX').

<u>COMMON BLOCK</u>	<u>VARIABLES USED</u>
BLOCKA	NFLAG
BLOCKD	RC, IARGS, NROW, NARGS, KIND
BLOCKE	L2 = 1 M(XAX') = 2 M(X'AX)
SCRAT	A

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
15		There must be between 8 and 10 arguments.
19-21		All arguments must be integers.
27		For the nine-argument form, one extra argument is given. Check that it is consistent.

TRANSF (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
29	200	For the ten argument form, two extra arguments must be checked.
36-46	240-260	Expand the eight-argument form into the equivalent ten-argument form.
48-50	280,300	Expand the nine-argument form into the equivalent ten-argument form.
51,52	320	Set arguments 11 and 12 to the implied dimensions of the result.
53-55		Check that all three matrices fit in the worksheet.
64-66		Set constants for executing $M(X'AX)$.
68-70	80	Set constants for executing $M(XAX')$.
71	90	Check that the dimension of the result does not exceed an allowable limit.
74-88	95-120	Perform the first matrix multiplication.
89-101	150-180	Perform the second matrix multiplication.
102-115	800,820	Place the resulting matrix into the worksheet.

VARCON(NAME)

PURPOSE: This subroutine checks to see if a name is one of the user controlled variables. ARG will indicate which variable or will be set to 0 if the name does not match.

COMMON BLOCK

BLOCKA

VARIABLES USED

ARG

VECTOR(A,J)

PURPOSE: This subroutine takes the value A and stores it in NRMAX successive locations in the worksheet starting at RC(J).

COMMON BLOCK

BLOCKD

VARIABLES USED

RC, NRMAX

WORKD(*)

PURPOSE: This subroutine is used to display worksheet sections.

COMMON BLOCK

BLOCKD

BLANK

VARIABLES USED

RC

KEY

WORKD(*) (cont.)

<u>LINE NUMBER</u>	<u>FORTTRAN LABEL</u>	<u>COMMENTS</u>
10		This arithmetic assignment subroutine is used to obtain the address of the element in the I'th row and J'th column of the worksheet.
12	1	KEM is the worksheet section to be displayed.
13-17	80	Erase the screen and write out a heading on the screen.
18-20		IA is the top row on the screen and IB is the bottom row. If KEY is less than 10, a top section is displayed. Otherwise, a lower section is displayed.
21,22		JA and JB are the numbers of the columns which will appear on the left (JA) and on the right (JB) of the screen. Only five columns are displayed at one time.
23-30	8,84	Write out the column numbers.
31-45	10,85,9	Display the worksheet section. The contents are dispatched to the screen eight rows at a time.

XECUTE

PURPOSE: This subroutine passes control to the subroutine in which the current command is to be executed.

COMMON BLOCK

BLOCKA

BLOCKB

BLANK

VARIABLES USED

NEWCD, NEWCDS, KSAVE, NSAVE, NFLAG

L1, L2

KEY, IOVLY

LINE
NUMBERFORTTRAN
LABELCOMMENTS

10,11

90

The value of L1 determines which subroutine to call to execute a given command.

12-68

100-3000

Various subroutines are called to execute given commands.

70,71

9001

The command is stored in the array NEWCDS.

72

Five commands are stored in NEWCDS before being stored in a data set.

73

There is room for forty records in the data set.

78

Notify the user that the data set has been filled.

XOMNIT

PURPOSE: This subroutine initializes several variables in the system.

COMMON BLOCKVARIABLES USED

BLOCKA

MODE, KSAVE

BLOCKD

NRMAX

BLANK

IOVLY

XPND(T,K,Y,KND)

PURPOSE: This subroutine is used during the compiling of an entered line to obtain arguments when asterisks have been used. T contains the information concerning the nature of the argument. Y will be set to the value of the indicated argument. K will be set to 0 if determining Y required only one element of T, and to 1 if two elements were required. A negative K indicates an error. KND will be used to distinguish floating point arguments from integer arguments.

COMMON BLOCKVARIABLES USED

BLOCKD

RC, IARGS, KIND, NRMAX, NROW, VWXYZ

LINE
NUMBER

FORTTRAN
LABEL

COMMENTS

15

IT will be used to determine what kind
of argument is being used.

XPND(T,K,Y,KND) (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
20		The argument is a worksheet reference. Transform IT so that it will represent the row.
21		Check that the row is within the limits of the worksheet.
24-27	41	The second element of T contains the column number. J will point to the head of the column.
30	46	J now points to the desired argument.
31,32		KND will be 0 if T(2) is positive and 1 if T(2) is negative.
33		Set Y to the desired value. J-1 will point to the correct location in the worksheet.
34		Set K to 1 to indicate that two values from T were needed.
39-40	60	IU will indicate which variable is referred to and KND will indicate its type.
41		Set K to 0 to indicate that only one value from T was needed.
43		Pick up the required argument from the VWXYZ array.

XPND(T,K,Y,KND) (cont.)

<u>LINE NUMBER</u>	<u>FORTRAN LABEL</u>	<u>COMMENTS</u>
45	70	The required argument is NRMAX.

CHAPTER IV

THE ADDITION OF COMMANDS

One can add commands to interactive OMNITAB in several ways. In many cases the new command can be embedded within an existing subprogram. In other cases, a new subprogram must be written. Both methods will be demonstrated.

A specific example of the first method mentioned above will be illustrated by the addition of the TRACE operation before the general procedure is outlined. When adding a command to interactive OMNITAB, one must consider what form the command should take. Trace is a matrix command. Hence, the key word should be MTRACE. (The prefix M indicates that this command is part of the set of matrix operations.) Two additional items of information are necessary for determining the trace of a matrix. The user must identify the matrix and indicate where its trace, the single scalar quantity, is to be stored. A matrix can be identified using four arguments (coordinates in the worksheet where it begins, number of rows, and number of columns), and its trace can be stored in a designated coordinate requiring two additional arguments. If the number of rows is different from the number of columns, the lesser of the two is taken as the order of the matrix of which the trace is to be calculated. Thus the command MTRACE requires, in general, six arguments. If only five are stated a regular trace will

be assumed, and the program will duplicate the third argument (number of rows.) The first two arguments will specify the upper left hand corner of the matrix. The last two will specify the coordinate where the trace is to be stored. The remaining argument(s) will specify the dimension of the matrix.

The MTRACE command has been added in the following steps:

- (1) Obtain the source program of SUBROUTINE LOOKUP (Appendix, page 225) and make the following changes
 - (a) The DIMENSION of MA must be increased from MA(18) to MA(20). (See comments below.)
 - (b) In DATA MA/ / the two integers 10035 and 815 must be added. (See explanation below.)
 - (c) The DO-loop starting at statement number 250 must be extended from 9 to 10.
- (2) Obtain the source program of SUBROUTINE MOP (Appendix, page 245)
 - (a) The computed GO TO statement (preceding 100) must be extended; after the last statement number (160), add a new one (e.g., 250).
 - (b) Three lines below statement number 105, insert (after J=1): IF (L2.EQ.10) J=2 and, about half-way between statements 105 and 110, insert: IF(L2.EQ.10) GO TO 260.
 - (c) Somewhere, after a closed section (an unconditional GO TO, or a RETURN) insert:

```

250 IARGS(7)=1
    IARGS(8)=1
    J=NARGS
    IF(NARGS.NE.5.AND.NARGS.NE.6) GO TO 10
    IF(NARGS.EQ.6) GO TO 105
    IARGS(6)=IARGS(5)
    IARGS(5)=IARGS(4)
    IARGS(4)=IARGS(3)
    GO TO 105

```

(d) Also add (after another closed branch)

```

260 TRACE=0.
    N=MINO(IARGS(3),IARGS(4))
    DO 270 NA=1,N
    TRACE=TRACE+RC(JB)
270 JB=JB+NROW+1
    ICX=IARGS(5)
    RC(ICX)=TRACE
    RETURN

```

(3) Submit the following batch program:

```

//EXEC FORTGCL,PARM.FORT='MAP',PARM.LKED=(XREF,LET,OVLY),
                                                    REGION=160K
//FORT.SYSIN DD *

```

Here insert the source decks of all subprograms
that have been modified and any new subprograms
added; in this instance
SUBROUTINE LOOKUP
SUBROUTINE MOP

```

/*
    Here insert the "LOAD MODULE" JCL sequence,
    Figure IV-1, starting with
//LKED.SYSLIB DD DSN=SYS1.GRAPHLIB,DISP=SHR
    and ending with
INSERT LOOKUP
/*

```

FIGURE IV-1

3 pages

LOAD MODULE JCL SEQUENCE

```

//LKED.SYSLIB DD DSN=SYS1.GRAPHLIB,DISP=SHR
// DD DSN=SYS1.UGALIB,DISP=SHR
// DD DSN=SYS1.FORTLIB,DISP=SHR
// DD DSN=SYS1.SSPLIB,DISP=SHR
// DD DSN=SYS1.LINKLIB,DISP=SHR
// DD DSN=SYS1.GMSLIB,DISP=SHR
//LKED.SYSLMOD DD DSN=SYS1.GRAPHLIB(OMTAB),DISP=SHR
//                SPACE=(TRK,(1,0,1))
//LKED.SYSIN DD *
  INCLUDE SYSLIB(OMTAB)
  ENTRY MAIN
  OVERLAY ONE
  INSERT ASTER
  OVERLAY TWO
  INSERT NONBLA
  OVERLAY TWO
  INSERT PHYCON
  OVERLAY TWO
  INSERT VARCON
  OVERLAY ONE
  INSERT XECUTE
  OVERLAY TWO
  INSERT STATD
  OVERLAY THREE
  INSERT GAMP,CHIX,CHIP,CHIZ
  OVERLAY THREE
  INSERT DETAX,BETAP,FFX,PPP,TTX,TTP
  OVERLAY THREE
  INSERT DETAZ,FFZ,TTZ
  OVERLAY TWO
  INSERT MISC2
  OVERLAY TWO
  INSERT MOVE
  OVERLAY TWO
  INSERT PMOTE
  OVERLAY TWO
  INSERT MSCROW
  OVERLAY TWO
  INSERT PROROW

```

OVERLAY TWO
INSERT DEFINE
OVERLAY TWO
INSERT EXTREM
OVERLAY TWO
INSERT SOR DER
OVERLAY TWO
INSERT ERASE
OVERLAY TWO
INSERT EXCHNG
OVERLAY TWO
INSERT FLIP
OVERLAY TWO
INSERT CHANGE
OVERLAY TWO
INSERT MATRIX
OVERLAY TWO
INSERT MOP
OVERLAY TWO
INSERT INVERT, INVCHK, SPINV
OVERLAY TWO
INSERT MMULT
OVERLAY TWO
INSERT MRAISE
OVERLAY TWO
INSERT GENER
OVERLAY TWO
INSERT ARITH
OVERLAY TWO
INSERT MXTX
OVERLAY THREE
INSERT TRANSF
OVERLAY THREE
INSERT MDAMAD
OVERLAY THREE
INSERT ARYVEC
OVERLAY TWO
INSERT EXPCON
OVERLAY TWO
INSERT READX
OVERLAY TWO
INSERT RESET
OVERLAY TWO
INSERT SET
OVERLAY TWO
INSERT PUNCT
OVERLAY THREE
INSERT FSIN, FCOS
OVERLAY ONE

INSERT INPUT, OMCONV
OVERLAY ONE
INSERT DISPLY
OVERLAY ONE
INSERT WORKD
OVERLAY ONE
INSERT COMAND
OVERLAY ONE
INSERT XOMNIT
OVERLAY ONE
INSERT SETQ
OVERLAY ONE
INSERT READQ
OVERLAY ONE
INSERT LOOKUP

To understand, and apply analogously for other additions, the operations included in step 1, the programmer must recognize the functioning of the SUBROUTINE LOOKUP. One must add all new commands to the dictionary of existing commands contained in the SUBROUTINE LOOKUP. The subroutine compares the key word entered by the user with a list of available commands. Thus, before a new command can be used, it must be included in this list. To do this one must first understand the method used in interpreting a key word appearing in the reply area.

A key word may consist of up to six letters (with a blank indicating the end of the key word if it requires fewer than six letters). This string of letters is converted into two integer values which are stored as the first two elements of the array NAME contained in the labelled COMMON called BLOCKE. NAME(1) is determined from the first three letters of the key word and NAME(2) from the last three using the code shown in Table IV-1[15]. Thus, for MTRACE, NAME(1)=9477+540+19=10035 and NAME(2)=729+81+5=815. Thus, somewhere within LOOKUP, NAME(1) and NAME(2) must be compared with these values.

LOOKUP is logically divided into two main sections. The first section consists of a series of data statements which are used to define more than a dozen integer arrays. In the second section NAME(1) and NAME(2) are compared with these arrays to determine which command is to be executed. When the appropriate command is found, two variables, L1 and L2, which are also stored in BLOCKE, are set. These variables will be used by the subroutine XECUTE to determine the appropriate subroutine in which the command will be carried out. The commands are divided into several groups of related commands. Each

	First Letter	Second Letter	Third Letter
A	729	27	1
B	1458	54	2
C	2187	81	3
D	2916	108	4
E	3645	135	5
F	4374	162	6
G	5103	189	7
H	5832	216	8
I	6561	243	9
J	7290	270	10
K	8019	297	11
L	8748	324	12
M	9477	351	13
N	10206	378	14
O	10935	405	15
P	11664	432	16
Q	12393	459	17
R	13122	486	18
S	13851	513	19
T	14580	540	20
U	15309	567	21
V	16038	594	22
W	16767	621	23
X	17496	648	24
Y	18225	675	25
Z	18954	702	26

Table IV-1

Conversion Codes

group is identified by the value of L1 which currently takes on values from one through fourteen. L2 is used to distinguish the different commands within a particular group. The characteristics of each of the groups are described in the comments in the listing of LOOKUP (Appendix, page 225).

There are many places in which a new command can be positioned within LOOKUP. One may use one of the existing groups or one may create a new group. For example, one might choose to set L1=15 for a new command with entirely new structure. If this were done, then it would be necessary to modify the computed GO TO in XECUTE to allow a branch when L1=15. This branch would call the appropriate subroutine which must be supplied to perform the new command.

Before resorting to this extreme, however, one might consider placing the new command within one of the existing groups. For MTRACE it was decided that the group consisting of the commands MDEFINE, ADEFINE, AERASE, MIDENT, ADIAG, MDIAG, MZERO, AZERO and MERASE would be very appropriate for MTRACE. For this group, L1=7 (see LOOK173-175, Appendix, page 228). The corresponding array of numerical equivalents of command names is in array MA (see LOOK 39-44, Appendix, page 225) which now must be extended to hold two additional values (see steps 1 (a) to (c) above).

At this point, one has to look at the SUBROUTINE XECUTE. In this example no changes are necessary since MTRACE has been included in an existing group (MOP). This group of commands is identified by L1=7. From XECUTE one learns that the computed GO TO statement (XECU10-11, Appendix, page 288) points to statement No. 1500 if L1=7.

Statement 1500 (XECU27) calls SUBROUTINE MOP. Thus we must obtain the source of MOP which must be modified to accomodate the new command.

The first executable statement in MOP (Appendix, page 245) is a computed GO TO. The number of branches in this statement must be increased by one member (step 2(a)). Step (2c) adds the new code required to do the checking of arguments characteristic of MTRACE. The general layout of IARGS for any matrix command is as follows:

- IARGS(1) Row coordinate of first matrix
- (2) Col. coordinate of first matrix
- (3) No. of rows of first matrix
- (4) No. of cols. of first matrix
- 5-8 (same for second matrix)
- 9-12 (same for third matrix, if any)

In our example, the second matrix is output and is a scalar; hence IARGS(7) and IARGS(8) need to be set equal to 1. The remainder of the code in step (2c) deals with checks for legitimacy and the duplication of the number of rows into number of columns, if the latter is not stated. NARGS is set according to the number of arguments which the user entered. The correct number of arguments for MTRACE is either 6 or 5 (only 5 required if the trace of a square matrix was desired). If the user entered more than 6 or fewer than 5 arguments, the program proceeds to 10 which is a call to the error routine. If the user entered 6 arguments they can be assumed to be the proper arguments in the order required by IARGS (if they exceed dimensions of the worksheet or are otherwise illegal, the subroutine MTXCHK and other utility routines will perform the checks). The next 3 statements in 2(c) are

are intended to duplicate the third argument (no. of rows of the matrix) into the fourth, and shift the others over, if the user intends only 5 arguments.

Step 2(b) serves two functions. The first insert will serve to inform MTXCHK that it must check two matrices. J indicates the number of matrices involved. L2=10 points to the 10th member of group 7 (which is the one we added). The second forces a transfer to step 2(d) which, here as in many similar additions, is the actual execution of the new command. Step 2(d) is the execution phase. In 2(d) it is to be noted that RC is a one-dimensional array of 2439 words which contains the entire worksheet. JB has been calculated by MOP as being the coordinate of the initial element of the first matrix. The worksheet is stored columnwise (thus NROW=80). The subroutine MTXCHK, which was called earlier, converted the entries in IARGS(1) and (2) and those in IARGS(5) and (6) into a single number pointing to the number in the RC array where each matrix begins, and placed it into IARGS(1) or IARGS(5) respectively. Stop (3) indicates how a batch job is to be submitted, so that the load module may be rebuilt, in our installation.

During the discussion of the addition of the command MTRACE, it was mentioned that an entirely new program could be written. Since this interactive program is designed for use by statisticians it would be useful to be able to evaluate percentiles, integrals and ordinates for certain probability distributions. These routines were thus added to the system. To help a programmer who wishes to make similar additions, the steps taken for this modification are described below:

The first change to be made is in the subroutine LOOKUP (appendix, page 225). A new array IG must be introduced to accommodate the numerical equivalents of the 18 new commands included, hence IG(36) was added in LOOK4. This array will contain the values against which NAME(1) and NAME(2) can be compared when one of the statistical functions is desired. A DATA statement (LOOK 96) was added to set the number codes corresponding to the names into the IG array (see Table IV-1). Following statement number 324, (LOOK 231) a section has been added. This section sets L1 to 15 since this group of commands is now the fifteenth. NAME(1) and NAME(2) are compared with pairs of values from IG to determine if the given command is a member of this group. If it is, then L2 will indicate which member it is.

The next changes must be made in XECUTE. Since L1 can now be 15, another branch (3000) has been added to the computed GO TO at statement number 90. Following statement number 2700, (after XECU66) these two lines have also been added:

```
GO TO 9000
3000 CALL STATD
```

Having added a call for the subroutine STATD to the subroutine XECUTE, one next must write STATD. This task was relatively easy since the new commands can be treated analogously to those executed by the subroutine FUNCT (Appendix, page 214). Both groups of commands allow all arguments (except, of course, the last one which must be a column number) to be either floating point constants or column numbers. STATD references a number of function subprograms. These were obtained from a statistical distribution package developed by

Bargmann [1]. We used FUNCT as a model to guide us in writing STATD.

The program flow is discussed in Chapter III on page 116).

Finally a new load module must be created as outlined in step (3) above. The following cards were added to load module JCL following the INSERT XECUTE card.

```
OVERLAY TWO
INSERT STATD
OVERLAY THREE
INSERT GAMP,CHIX,CHIP,CHIZ
OVERLAY THREE
INSERT BETAX,BETAP,FFX,FFP,TTX,TTP
OVERLAY THREE
INSERT BETAZ,FFZ,TTZ
```

After these modifications were made the following new commands are available to the user: YORMX, YORMP, YORMZ, CHIX, CHIP, CHIZ, GAMP, GAMZ, TTX, TTP, TTZ, BETAX, BETAP, BETAZ, FFX, FFP and FFZ.

CHAPTER V

EXAMPLES

In this chapter, several examples of the use of an interactive OMNITAB version will be presented in detail to illustrate situations in which a statistician would derive help from such a system. These examples, in addition to illustrating the use of certain commands, should suggest other ways in which OMNITAB can be applied.

A. ORDER STATISTICS

Frequently, non-parametric analyses involve the ordering of data. One such technique is the Mann-Whitney U test [55]. In this section a program will be described which calculates the value of U needed for this test.

The first problem, of course, is to enter the data into the worksheet. Since the U test requires that the two samples be merged before sorting, it is convenient to read all the data into one column and to use a second column to indicate to which sample a particular observation belongs. Thus the first instruction could be

```
READ 1 2
```

A zero in column two will indicate one sample and a one will indicate the other.

After entering all of his data the user should check the worksheet [see Figure V-1] for errors before issuing the next command: SORT 1 2. In this command the first argument indicates the column to be sorted in its own field. Any additional arguments indicate columns containing concomitant information which will be carried along as rows are exchanged by the SORT command. Sorting is from low to high.

After sorting the data, the next step would be to attach ranks to the data. This is most easily done using the command: GENERATE 1. 1. *NRMAX* 3. This will cause numbers from 1. to NRMAX, the number of rows containing data, to be generated, using a step size of 1., in column 3.

At this point the user can take advantage of the ease with which data can be edited because he can view the worksheet. In this example it is necessary to search for ties in the data and to replace the ranks for tied data by the average of those ranks. Note that this is only necessary if the observations come from different samples. This can be done using the command: MDEFINE $k,3 n,1 a$ where k is the number of the first row in the tied group, n is the number of tied observations in the group and a is the average rank for the tied group. In this example, the user would begin the editing procedure, after viewing the worksheet, column 1 of Figure V-2 by issuing the command MDEFINE 17,3 5,1 19.0.

The other ties in the example were re-defined by

```
MDEFINE 24,3 5,1 26.0
MDEFINE 29,3 2,1 29.5
MDEFINE 31,3 3,1 32.0
MDEFINE 34,3 5,1 36.0
```

This produced Col. 3 of Figure V-2

OUTPUT AREA					
WORKSHEET PART 1					
C O L U M N S					
1	2	3	4	5	
1	49.0000	0.0	0.0	0.0	0.0
2	22.0000	0.0	0.0	0.0	0.0
3	0.0000	0.0	0.0	0.0	0.0
4	62.0000	0.0	0.0	0.0	0.0
5	48.0000	0.0	0.0	0.0	0.0
6	59.0000	0.0	0.0	0.0	0.0
7	64.0000	0.0	0.0	0.0	0.0
8	5.0000	0.0	0.0	0.0	0.0
9	89.0000	0.0	0.0	0.0	0.0
10	64.0000	0.0	0.0	0.0	0.0
11	67.0000	0.0	0.0	0.0	0.0
12	67.0000	0.0	0.0	0.0	0.0
13	59.0000	0.0	0.0	0.0	0.0
14	65.0000	0.0	0.0	0.0	0.0
15	69.0000	0.0	0.0	0.0	0.0
16	67.0000	0.0	0.0	0.0	0.0
17	59.0000	0.0	0.0	0.0	0.0
18	64.0000	0.0	0.0	0.0	0.0
19	66.0000	0.0	0.0	0.0	0.0
20	77.0000	0.0	0.0	0.0	0.0
21	3.0000	0.0	0.0	0.0	0.0
22	72.0000	0.0	0.0	0.0	0.0
23	75.0000	0.0	0.0	0.0	0.0
24	81.0000	0.0	0.0	0.0	0.0
25	58.0000	0.0	0.0	0.0	0.0
26	57.0000	0.0	0.0	0.0	0.0
27	28.0000	0.0	0.0	0.0	0.0
28	62.0000	0.0	0.0	0.0	0.0
29	56.0000	0.0	0.0	0.0	0.0
30	82.0000	0.0	0.0	0.0	0.0
31	33.0000	1.0000	0.0	0.0	0.0
32	77.0000	1.0000	0.0	0.0	0.0
33	83.0000	1.0000	0.0	0.0	0.0
34	70.0000	1.0000	0.0	0.0	0.0
35	78.0000	1.0000	0.0	0.0	0.0
36	80.0000	1.0000	0.0	0.0	0.0
37	75.0000	1.0000	0.0	0.0	0.0
38	80.0000	1.0000	0.0	0.0	0.0
39	66.0000	1.0000	0.0	0.0	0.0
40	70.0000	1.0000	0.0	0.0	0.0
REPLY AREA					

Figure V-1

Worksheet after reading in the data

OUTPUT AREA					
WORKSHEET PART 1					
C O L U M N S					
	1	2	3	4	5
1	3.00000	0.0	1.00000	0.0	0.0
2	5.00000	0.0	2.00000	0.0	0.0
3	6.00000	0.0	3.00000	0.0	0.0
4	22.0000	0.0	4.00000	0.0	0.0
5	29.0000	0.0	5.00000	0.0	0.0
6	33.0000	1.00000	6.00000	0.0	0.0
7	45.0000	1.00000	7.00000	0.0	0.0
8	48.0000	0.0	8.00000	0.0	0.0
9	49.0000	0.0	9.00000	0.0	0.0
10	52.0000	1.00000	10.0000	0.0	0.0
11	56.0000	0.0	11.0000	0.0	0.0
12	57.0000	0.0	12.0000	0.0	0.0
13	59.0000	0.0	13.0000	0.0	0.0
14	59.0000	0.0	14.0000	0.0	0.0
15	59.0000	0.0	15.0000	0.0	0.0
16	60.0000	0.0	16.0000	0.0	0.0
17	62.0000	0.0	18.0000	0.0	0.0
18	62.0000	0.0	19.0000	0.0	0.0
19	62.0000	1.00000	19.0000	0.0	0.0
20	62.0000	1.00000	19.0000	0.0	0.0
21	62.0000	1.00000	19.0000	0.0	0.0
22	63.0000	1.00000	22.0000	0.0	0.0
23	63.0000	1.00000	23.0000	0.0	0.0
24	64.0000	0.0	26.0000	0.0	0.0
25	64.0000	0.0	26.0000	0.0	0.0
26	64.0000	0.0	26.0000	0.0	0.0
27	64.0000	1.00000	26.0000	0.0	0.0
28	64.0000	1.00000	26.0000	0.0	0.0
29	65.0000	0.0	29.5000	0.0	0.0
30	65.0000	1.00000	29.5000	0.0	0.0
31	66.0000	0.0	32.0000	0.0	0.0
32	66.0000	1.00000	32.0000	0.0	0.0
33	66.0000	1.00000	32.0000	0.0	0.0
34	67.0000	0.0	36.0000	0.0	0.0
35	67.0000	0.0	36.0000	0.0	0.0
36	67.0000	0.0	36.0000	0.0	0.0
37	67.0000	1.00000	36.0000	0.0	0.0
38	67.0000	1.00000	36.0000	0.0	0.0
39	68.0000	1.00000	39.0000	0.0	0.0
40	68.0000	1.00000	40.0000	0.0	0.0
REPLY AREA					

Figure V-2

Worksheet after entering GENERATE 1. 1. *NRMAX* 3

At this point the user may take advantage of the designation of samples by zeros and ones in column 2. He is interested in the sum of the ranks for only one of the two samples and can obtain this with two more commands. First, he issues the command: `MULT 2 3 4` . This command will store, for i from one through $NRMAX$, the product of the i 'th element in column two and the i 'th element in column three as the i 'th element of column four. Thus the only non-zero elements of column four will be ranks for one of the samples. The next command would be `SUM 4 5` . This command sums the elements of column four and stores this sum in column five [see Figure V-3]. (Since OMNITAB is column-oriented, the result is written into all elements of column 5. Because of the instantaneous display on the graphics terminal, this causes no delay.) From this point, the rest of the calculation can, of course, be carried on in OMNITAB, but a desk calculator is sufficient. The most tedious task of obtaining the sum of the ranks for one group has been easily accomplished.

OUTPUT AREA
WORKSHEET PART 1

C O L U M N S

	1	2	3	4	5
1	3.00000	0.0	1.00000	0.0	1704.00
2	5.00000	0.0	2.00000	0.0	1704.00
3	6.00000	0.0	3.00000	0.0	1704.00
4	22.0000	0.0	4.00000	0.0	1704.00
5	29.0000	0.0	5.00000	0.0	1704.00
6	33.0000	1.00000	6.00000	6.00000	1704.00
7	45.0000	1.00000	7.00000	7.00000	1704.00
8	48.0000	0.0	8.00000	0.0	1704.00
9	49.0000	0.0	9.00000	0.0	1704.00
10	52.0000	1.00000	10.0000	10.0000	1704.00
11	56.0000	0.0	11.0000	0.0	1704.00
12	57.0000	0.0	12.0000	0.0	1704.00
13	59.0000	0.0	13.0000	0.0	1704.00
14	59.0000	0.0	14.0000	0.0	1704.00
15	59.0000	0.0	15.0000	0.0	1704.00
16	59.0000	0.0	16.0000	0.0	1704.00
17	62.0000	0.0	17.0000	0.0	1704.00
18	62.0000	0.0	18.0000	0.0	1704.00
19	62.0000	1.00000	19.0000	19.0000	1704.00
20	62.0000	1.00000	19.0000	19.0000	1704.00
21	62.0000	1.00000	19.0000	19.0000	1704.00
22	63.0000	1.00000	22.0000	22.0000	1704.00
23	63.0000	1.00000	23.0000	23.0000	1704.00
24	64.0000	0.0	26.0000	0.0	1704.00
25	64.0000	0.0	26.0000	0.0	1704.00
26	64.0000	0.0	26.0000	0.0	1704.00
27	64.0000	1.00000	26.0000	26.0000	1704.00
28	64.0000	1.00000	26.0000	26.0000	1704.00
29	65.0000	0.0	29.5000	0.0	1704.00
30	65.0000	1.00000	29.5000	29.5000	1704.00
31	66.0000	0.0	32.0000	0.0	1704.00
32	66.0000	1.00000	32.0000	32.0000	1704.00
33	66.0000	1.00000	32.0000	32.0000	1704.00
34	67.0000	0.0	36.0000	0.0	1704.00
35	67.0000	0.0	36.0000	0.0	1704.00
36	67.0000	0.0	36.0000	0.0	1704.00
37	67.0000	1.00000	36.0000	36.0000	1704.00
38	67.0000	1.00000	36.0000	36.0000	1704.00
39	68.0000	1.00000	39.0000	39.0000	1704.00
40	68.0000	1.00000	40.0000	40.0000	1704.00

REPLY AREA

Figure V-3

Worksheet after entering SUM 4 5

B. BIOASSAY

A frequent problem encountered by the applied statistician who works with biological data is that of estimation of parameters in non-linear models. A technique often used here is quantal analysis, usually called bioassay on account of its most frequent application. Many bioassay problems can be solved using OMNITAB. Convergence difficulties can be resolved as they appear. In this section, performance of a probit analysis using OMNITAB will be described.

For this example, consider the following experiment: A certain carcinogenic food additive is administered at several different dose levels to groups of laboratory mice. The experiment is continued for a fixed time period. At the end of this time period, the number of animals in each group (the i 'th group consists of all animals which have been administered dose d_i) which have either survived or have been eliminated following the discovery of a tumor are called the number at risk, n_i . For each group, the number of animals in which tumors have appeared is known as the number of responses, r_i .

Figure V-4 shows the statements which were used in performing the initial cycle. Data are read into columns 1, 2 and 3 (lines 2-10). Column 1 contains n_i , column 2 contains r_i and column 3 contains d_i . In the probit model, $Y_i = A + BX_i$, X_i is usually logdose while Y_i , the probit, is $\phi^{-1}(p_i) + 5$ where p_i is the proportion of successes in the i 'th group and is initially estimated by r_i/n_i , the observed proportion, and where ϕ is the proportion under the standard normal curve and thus ϕ^{-1} yields the percentile for the standard normal curve. Dosage (logdose) is obtained and stored in column 9 (line 11). The

OUTPUT AREA
IF THE SCREEN BECOMES FULL AN ALARM WILL SOUND. WHEN YOU WANT TO SEE
THE NEXT SECTION OF YOUR PROGRAM, PRESS KEY 2.

```

ERASE
READ 1 2 3
100 9 10
75 5 25
85 31 100
95 58 200
105 78 500
100 91 700
90 07 900
125 124 1000
LOCATE 3 9 LOCPOSE X
DIVIDE 2 1 4 OBSERVED PROPORTION
YORHP 4 5
ADD 5. 5 5 OBSERVED PROBIT
ADD 5 10 10 WORKING PROBIT Y
ADD 1 0 0 HEIGHTS H
SUM 0 15 SUM OF HEIGHTS
MULT 9 10 11 XY
MULT 9 9 12 XX
DIVIDE 1.9 8.4 0 10.7 HEIGHTED SUMS
DIVIDE 10.7 15 10.7 26
SUBTRACT 10.10 20 26 SXN
DIVIDE 10.7 15 10.0 27
SUBTRACT 10.9 27 27 SXY
DIVIDE 27 26 21 B
DIVIDE 10.7 15 13 XBAR
DIVIDE 10.0 15 14 YBAR
MULT 21 13 10
SUBTRACT 14 10 10 A
MULT 9 21 6
ADD 6 10 0 PREDICTED PROBIT
SUBTRACT 6 5. 26
YORHP 20 7 PREDICTED PROPORTION
YORHP 20 27 2
SUBTRACT 4 7 10
DIVIDE 10 27 10
ADD 6 10 16
SUBTRACT 1. 7 0
MULT 1 7 0
DIVIDE 27 0 8
MULT 27 6 3 HEIGHT
MULT 0 1 0
SUM 0 10 SUM OF HEIGHTS

```

REPLY AREA

Figure V-4

First page of instructions used in bioassay example

observed proportion and the observed probit are calculated and stored in columns 4 and 5 respectively (lines 12-14). (YORMP evaluates the function Φ^{-1} .) These values need not be recalculated later. The worksheet, after these instructions have been completed, appears as Figure V-5 and can be used to compare with predicted proportions later.

After completion of this initial work, weighted regression is initiated. Column 10 will contain the working probit. For the initial cycle, the working probit is the observed probit (line 15). Column 8 will contain the weights which initially are the number at risk, n_i , contained in column 1 (line 16). Next a weighted regression was performed using as X, the logdose, and as Y, the working probit (lines 17-29).¹ The initial estimates, B(1.99483) and A(.83249), are stored in columns 21 and 16 respectively. These two sections of the worksheet (Figures V-6 and V-7) will be used to store the estimates of the slope and intercept obtained in the iterative procedure. Finally, the predicted probit, $Y = A + BX$, and the predicted proportion, $P = \Phi[Y-5]$, where Φ is the standard normal CDF, are calculated and stored in columns 6 and 7 respectively (lines 30-33). (YORMX evaluates the function Φ .) Figures V-8 and V-9 show the worksheet sections 3 and 4 as they appeared at the end of the first cycle.

In subsequent cycles working probits, y^* , and weights, w , must be calculated as follows: $y^* = Y + \frac{p-P}{Z}$ and $w = \frac{nZ^2}{(p(1-p))}$, where Y is the predicted probit, p is the observed proportion, P is the predicted

¹Where more than the required arguments appear in a column-oriented OMNITAB command, the next to last column (or number) is multiplied; e.g., line 21 says take number in (10,7) divide by entries in column 15, multiply result by number in (10,7) and store into Col. 16.

OUTPUT AREA
WORKSHEET PART 1

C O L U M N S

	1	2	3	4	5
1	100.000	3.00000	10.0000	0.300000E-01	3.11921
2	75.0000	5.00000	25.0000	0.600000E-01	3.49691
3	50.0000	31.0000	100.000	0.304706	4.65409
4	95.0000	59.0000	200.000	0.610520	5.20009
5	105.000	75.0000	500.000	0.742937	5.65210
6	100.000	91.0000	700.000	0.910000	6.34075
7	90.0000	07.0000	900.000	0.955687	6.83391
8	125.000	124.000	1000.00	0.932000	7.40691
9	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0
36	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0

REPLY AREA

Figure V-5

Worksheet after entering YORMP 4 5

OUTPUT AREA
WORKSHEET PART 4

C O L U M N S

	16	17	18	19	20
1	0.032488	0.0	0.0	0.0	0.0
2	0.032488	0.0	0.0	0.0	0.0
3	0.032488	0.0	0.0	0.0	0.0
4	0.032488	0.0	0.0	0.0	0.0
5	0.032488	0.0	0.0	0.0	0.0
6	0.032488	0.0	0.0	0.0	0.0
7	0.032488	0.0	0.0	0.0	0.0
8	0.032488	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0
36	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0

REPLY AREA

Figure V-6

Worksheet after entering DIVIDE 27 26 21 B

OUTPUT AREA					
WORKSHEET PART 5					
C O L U M N S					
	21	22	23	24	25
1	1.99483	0.0	0.0	0.0	0.0
2	1.99483	0.0	0.0	0.0	0.0
3	1.99483	0.0	0.0	0.0	0.0
4	1.99483	0.0	0.0	0.0	0.0
5	1.99483	0.0	0.0	0.0	0.0
6	1.99483	0.0	0.0	0.0	0.0
7	1.99483	0.0	0.0	0.0	0.0
8	1.99483	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0
36	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0
REPLY AREA					

Figure V-7

Worksheet after entering SUBTRACT 14 16 16 A

OUTPUT AREA
WORKSHEET PART 2

C O L U M N S					
	6	7	8	9	10
1	2.82732	0.149022E-01	100.000	1.00000	3.11921
2	3.62115	0.838697E-01	75.0000	1.33794	3.43891
3	4.02218	0.423423	65.0000	2.00000	4.65409
4	5.42268	0.683728	95.0000	2.30103	5.28069
5	6.21649	0.863100	105.000	2.69897	5.65218
6	6.50799	0.934221	100.000	2.84510	6.34075
7	6.72571	0.957800	90.0000	2.95424	6.63391
8	6.81699	0.965991	125.000	3.00000	7.40891
9	0.0	0.0	0.0	0.0	0.0
10	0.0	1002.23	4240.32	10025.4	4574.37
11	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0
36	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0

REPLY AREA

Figure V-8

Worksheet after entering YORMX 26.7 PREDICTED PROPORTION

OUTPUT AREA
WORKSHEET PART 3

C O L U M N S

	11	12	13	14	15
1	3.11921	1.00000	2.32545	5.47138	775.000
2	4.89127	1.95424	2.32545	5.47138	775.000
3	9.30818	4.00000	2.32545	5.47138	775.000
4	12.1510	5.29473	2.32545	5.47138	775.000
5	15.2551	7.28444	2.32545	5.47138	775.000
6	18.0401	6.09480	2.32545	5.47138	775.000
7	20.1890	0.72753	2.32545	5.47138	775.000
8	22.2267	9.00000	2.32545	5.47138	775.000
9	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0
36	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0

REPLY AREA

Figure V-9

Worksheet after entering DIVIDE *10,S* 15 14 YBAR

proportion and Z is the ordinate under the normal curve (YORMX) (lines 34-42). In each cycle, new estimates A and B are calculated by performing a weighted regression using as X , the logdose, and as Y , the working probit. Note that the working probits and the weights are recalculated for each cycle. This process can be continued until the desired degree of convergence is obtained.

C. SCALING OF MULTIDIMENSIONAL CATEGORIZED VARIABLES.

In the transformation of categorized (nominal) variables into interval scales [26,41,42] a cumbersome numerical analysis problem arises when three or more variables are categorized. One of the problems is as follows: Given k sets of random variables, with p_1, p_2, \dots, p_k random variables in each set, find vectors of weights $\underline{x}_1, \dots, \underline{x}_k$, and hence a single linear composite for each set: $u_1 = \underline{x}_1' \underline{y}_1, \dots, u_k = \underline{x}_k' \underline{y}_k$, such that the determinant of the matrix of correlations between (u_1, u_2, \dots, u_k) is minimum [56].

If close starting values can be found, the minimizing sets of weights, $\underline{x}_1, \underline{x}_2, \dots, \underline{x}_k$ can be obtained, e.g., by Fletcher-Powell iteration [27]. Thus we need to investigate techniques of approximation and since scaling of categorized data is in itself a very crude method of data reduction, the approximate solution may be quite adequate in lieu of the exact minimizing solution, if it is close enough.

Many methods are available and are being studied [16]. One could be to obtain the canonical weights of the i 'th set against all other sets combined. Another would be to obtain the canonical weights of set i versus each of the other sets, and then to combine those "images"

by some weighting procedure (multiple regression of suitably normalized weights, simple averages, etc.) Clearly, OMNITAB is very useful for exploration of these various weighting methods. The present illustration compares images that were averaged using canonical correlations as weights. The first two frames (Figures V-10 and V-11) show the commands which were used to obtain all the canonical variables, and the combined linear composite for the first set. The following two frames (Figures V-12 and V-13) show the pages of the worksheet which contain results. The entire correlation matrix, of order six by six (three sets of two variables each), was read into the worksheet starting in location (1,1); thus R_{12} starts at (1,3), R_{13} at (1,5), and R_{23} at (3,5). (10,1) is the start of the 2×2 matrix $R_{12}R_{12}'$, (13,1) of $R_{13}R_{13}'$, (16,1) of $R_{23}R_{23}'$, (19,1) of $(R_{12}R_{12}')^{-1}$, (22,1) of $(R_{13}R_{13}')^{-1}$, (25,1) of $(R_{23}R_{23}')^{-1}$. The inverses, in some applications, are singular or near-singular, if this occurs, certain variables need to be dropped. On the basis of displays in Figure V-12 we observe that the matrices are well-conditioned and in the present instance, we may proceed. The corresponding rows in columns four and five contain check procedures for eigenvectors. The first two columns of row 28 contain the eigenvector associated with the largest root of $R_{12}R_{12}'$, hence (since $R_{11} = R_{22} = I$) the image of set 2 in set 1 (unit length). Similarly row 30 contains the image of set 3 in set 1 and row 32 contains the image of set 3 in set 2. The 2 by 2 matrix of correlations between images starts at (10,6) (on the second page of the worksheet.) As is seen in Figure V-12, the two row vectors in rows 28 and 30 are quite different. It is on the basis of this worksheet display that the decision was made to combine these vectors

OUTPUT AREA
IF THE SCREEN BECOMES FULL AN ALARM WILL SOUND. WHEN YOU WANT TO SEE
THE NEXT SECTION OF YOUR PROGRAM, PRESS KEY 2.

```

ERASE
READ 1MM6
1.0..01737..10028..03599..03898
0.1..-33296..07394..-07884..-02135
.01737..-33296.1.0..-15750..-14852
.16088..07394.0.1..16727..00848
.03599..-07884..-15750..16727.1.0
.03898..-02135..-14852..00848.0.1
MIXX 1 1.3.2.2.10.1
MIXX 1 1.5.2.2.13.1
MIXX 1 3.5.2.2.16.1
GENERATE 1..0..1..30
RESET NMAX 6
GENERATE 1..0..1..30
ADD 1. TO 30.30
GENERATE 1..0..1..30
HVECDIAG 10.1.2.2.10.4
HVECDIAG 13.1.2.2.13.4
HVECDIAG 16.1.2.2.16.4
MINVERT 10.1.2.19.1
MINVERT 13.1.2.22.1
MINVERT 16.1.2.25.1
READ 1.2
ROW 20
.06717476..93774122
READ 4
.116725415
READ 4
.160880
ROW 20
.116725415
MMULT 20.1.1.2.10.1.2.2.19.4
SUM 4 ROW 13 TO ROW 14 STORE 8
READ 1.2
ROW 30
.5171150342..-055916425
MMULT 30.1.1.2.13.1.2.2.22.4
MSCALAR 22.4.1.2.112.50298.23.4
MSCALAR 19.4.1.2.0.5671145397.20.4
READ 1.2
ROW 32
-.013261867..50189707364
MMULT 32.1.1.2.16.1.2.2.25.4

```

REPLY AREA

Figure V-10

First page of commands for scaling example

```

                                OUTPUT AREA
MSCALAR 25.4.1.2.15.01147052733.26.4
READ 1.2
ROW 32
-.01318073..58200008
MMULT 32.1.1.2.16.1.2.2.25.4
MSCALAR 25.4.1.2.15.01028849.26.4
MSCALAR 29.1.1.2.-1..29.1
MERASE 1.0.29.1
MTRANS 30.1.1.2.29.4
MMULT 29.1.1.2.29.4.2.1.31.4
READ 0.7.0
ROW 10
1..019245..34165
.019245.1..09428
MINVERT 10.6.2.13.6
MMULT 13.6.2.2.10.8.2.1.19.8
MSCALAR 29.1.1.2..804079.29.1
MSCALAR 30.1.1.2..564458.31.1
MADD 29.1.1.2.31.1.34.1
READ 9.10
-.734974..670095
MSCALAR 29.1.1.2..10.0..29.1
MSCALAR 30.1.1.2..11.0..31.1
MADD 29.1.1.2.31.1.34.1
READ 9.10
-.061009..9981321
ROW 21

```

REPLY 8868

Figure V-11

Second page of commands for scaling example

OUTPUT AREA
WORKSHEET PART 1

	C O L U M N S				
	1	2	3	4	5
1	1.00000	0.0	0.173700E-01	0.160680	0.359900E-01
2	0.0	1.00000	-0.332960	0.738400E-01	-0.788400E-01
3	0.173700E-01	-0.332960	1.00000	0.0	-0.157500
4	0.160680	0.738400E-01	0.0	1.00000	0.167270
5	0.359900E-01	-0.788400E-01	-0.157500	0.167270	1.00000
6	0.389900E-01	-0.213500E-01	-0.148520	0.848000E-02	0.0
7	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0
10	0.261041E-01	0.609596E-02	0.0	0.261841E-01	0.0
11	0.609596E-02	0.116315	0.0	0.116315	0.0
12	0.0	0.0	0.0	0.0	0.0
13	0.281472E-02	-0.366967E-02	0.0	0.281472E-02	0.0
14	-0.366967E-02	0.667156E-02	0.0	0.667156E-02	0.0
15	0.0	0.0	0.0	0.0	0.0
16	0.468644E-01	-0.276046E-01	0.0	0.468644E-01	0.0
17	-0.276046E-01	0.280511E-01	0.0	0.280511E-01	0.0
18	0.0	0.0	0.0	0.0	0.0
19	38.6628	-2.02626	0.0	0.704099E-02	0.116461
20	-2.02626	8.70356	0.0	0.671746E-01	0.997737
21	0.0	0.0	0.0	0.0	0.0
22	1255.93	690.019	0.0	0.469046E-02	-0.760793E-02
23	690.019	629.073	0.0	0.517115	-0.865915
24	0.0	0.0	0.0	0.0	0.0
25	60.7628	49.9542	0.0	-0.541754E-01	0.387794E-01
26	49.9542	84.0079	0.0	-0.813108	0.682000
27	0.0	0.0	0.0	0.0	0.0
28	-0.671747E-01	-0.997741	0.0	0.517116	0.0
29	-0.229502E-01	-0.340878	0.0	-0.055913	0.0
30	0.517116	-0.865915	0.0	0.0	0.0
31	0.407596E-01	-0.808980E-01	0.0	0.019245	0.0
32	-0.013108	0.992000	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0
34	0.256034E-01	-0.421574	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0
36	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0

REPLY AREA

Figure V-12

Worksheet after entering MADD 29,1,1,2,31,1,34,1

OUTPUT AREA					
WORKSHEET PART 2					
C O L U M N S					
	6	7	8	9	10
1	0.988800E-01	0.0	0.0	-0.610930E-01	0.998132
2	-0.213500E-01	0.0	0.0	0.0	0.0
3	-0.145520	0.0	0.0	0.0	0.0
4	0.848000E-02	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0
6	1.00000	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0
10	1.00000	0.818245	0.341650	0.0	0.0
11	0.818245	1.00000	0.942799E-01	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	3.04101	-2.49133	0.804079	0.0	0.0
14	-2.49133	3.04101	-0.564458	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0
36	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0
REPLY AREA					

Figure V-13

Worksheet after entering in columns 9 and 10 the values
.061093 and .9981321

using some systematic weighting procedure (had the vectors been similar, different weighting procedures would have had little effect). With canonical correlations used as weights (10,8) and (11,8), the two images (28,1) and (30,1) have been combined, then normalized into the 1 by 2 vector starting at (1,9).

The next frame (Figure V-14) shows the additional commands which were used to obtain images of set 2 in set 1 ($R_{12}^1 x_1$, reduced to unit length), of set 3 in set 1, and of set 3 in set 2. After combination with canonical correlations as weights, and normalizations, the final weights were obtained and recorded in the second part of the worksheet display starting at (1,9)

(see Figure V-15). These weights $\begin{bmatrix} -.06109 & .99813 \\ .91737 & -.39803 \\ .87288 & .45992 \end{bmatrix}$ are so

comfortably close to the final minimum-determinant solution (which was obtained by Fletcher-Powell iteration).

$\begin{bmatrix} -.05466 & .99851 \\ .92930 & -.36932 \\ .85659 & .51599 \end{bmatrix}$ that this approximation, if equally close in

many other examples, can well be considered adequate for purposes of categorical scaling. The correlations between the u_1, u_2, u_3 were obtained by MMULT commands, e.g., $x_1^1 R_{12}$ into (5,9) and $x_1^1 R_{12} x_2 = \text{corr}(u_1, u_2)$ into (10,10). (see Figure V-16). The correlation matrix

$\begin{bmatrix} 1 & -.331275 & -.0815043 \\ -.331275 & 1 & -.248449 \\ -.081504 & -.248449 & 1 \end{bmatrix}$ is quite close to the

minimum-determinant matrix

$\begin{bmatrix} 1 & -.338552 & -.075643 \\ -.338552 & 1 & -.251130 \\ -.075643 & -.251130 & 1 \end{bmatrix}$

```

                                OUTPUT AREA
MSCPLAR 25.4.1.2.15.01147052733.26.4
READ 1.2
ROW 32
-.61318073..59200008
MMULT 32.1.1.2.16.1.2.2.25.4
MSCALAR 25.4.1.2.15.01026848.26.4
MSCALAR 28.1.1.2.-1.29.1
MERISE 1.0.20.1
MTRANS 30.1.1.2.28.4
MMULT 28.1.1.2.28.4.2.1.31.4
READ 6.7.8
ROW 10
1..019245..34165
.019245.1..09420
MINVERT 10.6.2.13.6
MMULT 13.6.2.2.10.8.2.1.13.8
MSCALAR 28.1.1.2..804079.29.1
MSCALAR 30.1.1.2.-.564458.31.1
MADD 29.1.1.2.31.1.34.1
READ 9.10
-.734974..-670095
MSCALAR 28.1.1.2..10.8..29.1
MSCALAR 30.1.1.2..11.8..31.1
MADD 29.1.1.2.31.1.34.1
READ 9.10
-.061093..9981321
MMULT 28.1.1.2.1.3.2.2.28.4
READ 6.7
ROW 16
.9809464..-2472700
.013189..-582
.34165..25811
MMULT 18.6.1.2.16.6.2.2.20.6
READ 9.10
ROW 2
.917373..-390030
MMULT 30.1.1.2.1.5.2.2.30.4
MMULT 32.1.1.2.3.5.2.2.32.4
READ 9.10
ROW 10
.91314043..40762721
.0733795..40704036
.0942787..2581134
MMULT 18.6.1.2.16.6.2.2.20.9
READ 9.10
ROW 3

```

```

                                REPLY AREA

```

Figure V-14

Second page of commands for scaling example

OUTPUT AREA				
WORKSHEET PART 2				
C O L U M N S				
	6	7	8	9
1	0.999800E-01	0.0	0.0	-0.610930E-01
2	-0.213500E-01	0.0	0.0	0.917373
3	-0.148520	0.0	0.0	0.872076
4	0.848000E-02	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0
6	1.00000	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0
10	1.00000	0.019245	0.341650	0.0
11	0.019245	1.00000	0.942799E-01	0.0
12	0.0	0.0	0.0	0.0
13	3.04101	-2.49133	0.004079	0.0
14	-2.49133	3.04101	-0.564458	0.0
15	0.0	0.0	0.0	0.0
16	0.968946	-0.247271	0.0	0.913148
17	0.013189	-0.502000	0.0	0.679379
18	0.941050	0.258110	0.0	0.042799E-01
19	0.0	0.0	0.0	0.0
20	0.540939	-0.234700	0.0	0.311520
21	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0
36	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0
41	0.0	0.0	0.0	0.0
42	0.0	0.0	0.0	0.0
43	0.0	0.0	0.0	0.0
44	0.0	0.0	0.0	0.0
45	0.0	0.0	0.0	0.0
46	0.0	0.0	0.0	0.0
47	0.0	0.0	0.0	0.0
48	0.0	0.0	0.0	0.0
49	0.0	0.0	0.0	0.0
50	0.0	0.0	0.0	0.0
51	0.0	0.0	0.0	0.0
52	0.0	0.0	0.0	0.0
53	0.0	0.0	0.0	0.0
54	0.0	0.0	0.0	0.0
55	0.0	0.0	0.0	0.0
56	0.0	0.0	0.0	0.0
57	0.0	0.0	0.0	0.0
58	0.0	0.0	0.0	0.0
59	0.0	0.0	0.0	0.0
60	0.0	0.0	0.0	0.0
61	0.0	0.0	0.0	0.0
62	0.0	0.0	0.0	0.0
63	0.0	0.0	0.0	0.0
64	0.0	0.0	0.0	0.0
65	0.0	0.0	0.0	0.0
66	0.0	0.0	0.0	0.0
67	0.0	0.0	0.0	0.0
68	0.0	0.0	0.0	0.0
69	0.0	0.0	0.0	0.0
70	0.0	0.0	0.0	0.0
71	0.0	0.0	0.0	0.0
72	0.0	0.0	0.0	0.0
73	0.0	0.0	0.0	0.0
74	0.0	0.0	0.0	0.0
75	0.0	0.0	0.0	0.0
76	0.0	0.0	0.0	0.0
77	0.0	0.0	0.0	0.0
78	0.0	0.0	0.0	0.0
79	0.0	0.0	0.0	0.0
80	0.0	0.0	0.0	0.0
81	0.0	0.0	0.0	0.0
82	0.0	0.0	0.0	0.0
83	0.0	0.0	0.0	0.0
84	0.0	0.0	0.0	0.0
85	0.0	0.0	0.0	0.0
86	0.0	0.0	0.0	0.0
87	0.0	0.0	0.0	0.0
88	0.0	0.0	0.0	0.0
89	0.0	0.0	0.0	0.0
90	0.0	0.0	0.0	0.0
91	0.0	0.0	0.0	0.0
92	0.0	0.0	0.0	0.0
93	0.0	0.0	0.0	0.0
94	0.0	0.0	0.0	0.0
95	0.0	0.0	0.0	0.0
96	0.0	0.0	0.0	0.0
97	0.0	0.0	0.0	0.0
98	0.0	0.0	0.0	0.0
99	0.0	0.0	0.0	0.0
100	0.0	0.0	0.0	0.0
101	0.0	0.0	0.0	0.0
102	0.0	0.0	0.0	0.0
103	0.0	0.0	0.0	0.0
104	0.0	0.0	0.0	0.0
105	0.0	0.0	0.0	0.0
106	0.0	0.0	0.0	0.0
107	0.0	0.0	0.0	0.0
108	0.0	0.0	0.0	0.0
109	0.0	0.0	0.0	0.0
110	0.0	0.0	0.0	0.0
111	0.0	0.0	0.0	0.0
112	0.0	0.0	0.0	0.0
113	0.0	0.0	0.0	0.0
114	0.0	0.0	0.0	0.0
115	0.0	0.0	0.0	0.0
116	0.0	0.0	0.0	0.0
117	0.0	0.0	0.0	0.0
118	0.0	0.0	0.0	0.0
119	0.0	0.0	0.0	0.0
120	0.0	0.0	0.0	0.0
121	0.0	0.0	0.0	0.0
122	0.0	0.0	0.0	0.0
123	0.0	0.0	0.0	0.0
124	0.0	0.0	0.0	0.0
125	0.0	0.0	0.0	0.0
126	0.0	0.0	0.0	0.0
127	0.0	0.0	0.0	0.0
128	0.0	0.0	0.0	0.0
129	0.0	0.0	0.0	0.0
130	0.0	0.0	0.0	0.0
131	0.0	0.0	0.0	0.0
132	0.0	0.0	0.0	0.0
133	0.0	0.0	0.0	0.0
134	0.0	0.0	0.0	0.0
135	0.0	0.0	0.0	0.0
136	0.0	0.0	0.0	0.0
137	0.0	0.0	0.0	0.0
138	0.0	0.0	0.0	0.0
139	0.0	0.0	0.0	0.0
140	0.0	0.0	0.0	0.0
141	0.0	0.0	0.0	0.0
142	0.0	0.0	0.0	0.0
143	0.0	0.0	0.0	0.0
144	0.0	0.0	0.0	0.0
145	0.0	0.0	0.0	0.0
146	0.0	0.0	0.0	0.0
147	0.0	0.0	0.0	0.0
148	0.0	0.0	0.0	0.0
149	0.0	0.0	0.0	0.0
150	0.0	0.0	0.0	0.0
151	0.0	0.0	0.0	0.0
152	0.0	0.0	0.0	0.0
153	0.0	0.0	0.0	0.0
154	0.0	0.0	0.0	0.0
155	0.0	0.0	0.0	0.0
156	0.0	0.0	0.0	0.0
157	0.0	0.0	0.0	0.0
158	0.0	0.0	0.0	0.0
159	0.0	0.0	0.0	0.0
160	0.0	0.0	0.0	0.0
161	0.0	0.0	0.0	0.0
162	0.0	0.0	0.0	0.0
163	0.0	0.0	0.0	0.0
164	0.0	0.0	0.0	0.0
165	0.0	0.0	0.0	0.0
166	0.0	0.0	0.0	0.0
167	0.0	0.0	0.0	0.0
168	0.0	0.0	0.0	0.0
169	0.0	0.0	0.0	0.0
170	0.0	0.0	0.0	0.0
171	0.0	0.0	0.0	0.0
172	0.0	0.0	0.0	0.0
173	0.0	0.0	0.0	0.0
174	0.0	0.0	0.0	0.0
175	0.0	0.0	0.0	0.0
176	0.0	0.0	0.0	0.0
177	0.0	0.0	0.0	0.0
178	0.0	0.0	0.0	0.0
179	0.0	0.0	0.0	0.0
180	0.0	0.0	0.0	0.0
181	0.0	0.0	0.0	0.0
182	0.0	0.0	0.0	0.0
183	0.0	0.0	0.0	0.0
184	0.0	0.0	0.0	0.0
185	0.0	0.0	0.0	0.0
186	0.0	0.0	0.0	0.0
187	0.0	0.0	0.0	0.0
188	0.0	0.0	0.0	0.0
189	0.0	0.0	0.0	0.0
190	0.0	0.0	0.0	0.0
191	0.0	0.0	0.0	0.0
192	0.0	0.0	0.0	0.0
193	0.0	0.0	0.0	0.0
194	0.0	0.0	0.0	0.0
195	0.0	0.0	0.0	0.0
196	0.0	0.0	0.0	0.0
197	0.0	0.0	0.0	0.0
198	0.0	0.0	0.0	0.0
199	0.0	0.0	0.0	0.0
200	0.0	0.0	0.0	0.0
201	0.0	0.0	0.0	0.0
202	0.0	0.0	0.0	0.0
203	0.0	0.0	0.0	0.0
204	0.0	0.0	0.0	0.0
205	0.0	0.0	0.0	0.0
206	0.0	0.0	0.0	0.0
207	0.0	0.0	0.0	0.0
208	0.0	0.0	0.0	0.0
209	0.0	0.0	0.0	0.0
210	0.0	0.0	0.0	0.0
211	0.0	0.0	0.0	0.0
212	0.0	0.0	0.0	0.0
213	0.0	0.0	0.0	0.0
214	0.0	0.0	0.0	0.0
215	0.0	0.0	0.0	0.0
216	0.0	0.0	0.0	0.0
217	0.0	0.0	0.0	0.0
218	0.0	0.0	0.0	0.0
219	0.0	0.0	0.0	0.0
220	0.0	0.0	0.0	0.0
221	0.0	0.0	0.0	0.0
222	0.0	0.0	0.0	0.0
223	0.0	0.0	0.0	0.0
224	0.0	0.0	0.0	0.0
225	0.0	0.0	0.0	0.0
226	0.0	0.0	0.0	0.0
227	0.0	0.0	0.0	0.0
228	0.0	0.0	0.0	0.0
229	0.0	0.0	0.0	0.0
230	0.0	0.0	0.0	0.0
231	0.0	0.0	0.0	0.0
232	0.0	0.0	0.0	0.0
233	0.0	0.0	0.0	0.0
234	0.0	0.0	0.0	0.0
235	0.0	0.0	0.0	0.0
236	0.0	0.0	0.0	0.0
237	0.0	0.0	0.0	0.0
238	0.0	0.0	0.0	0.0
239	0.0	0.0	0.0	0.0
240	0.0	0.0	0.0	0.0
241	0.0	0.0	0.0	0.0
242	0.0	0.0	0.0	0.0
243	0.0	0.0	0.0	0.0
244	0.0	0.0	0.0	0.0
245	0.0	0.0	0.0	0.0
246	0.0	0.0	0.0	0.0
247	0.0	0.0	0.0	0.0
248	0.0	0.0	0.0	0.0
249	0.0	0.0	0.0	0.0
250	0.0	0.0	0.0	0.0
251	0.0	0.0	0.0	0.0
252	0.0	0.0	0.0	0.0
253	0.0	0.0	0.0	0.0
254	0.0	0.0	0.0	0.0
255	0.0	0.0	0.0	0.0
256	0.0	0.0	0.0	0.0
257	0.0	0.0	0.0	0.0
258	0.0	0.0	0.0	0.0
259	0.0	0.0	0.0	0.0
260	0.0	0.0	0.0	0.0
261	0.0	0.0	0.0	0.0
262	0.0	0.0	0.0	

.872876..458922
MMULT 1.9.1.2.1.9.2.2.5.9
MTRANS 1.9.9.2.22.6
MMULT 5.9.1.2.22.7.2.1.10.10
MMULT 1.9.1.2.1.5.2.2.6.9
MMULT 8.9.1.2.22.0.2.1.11.10
MMULT 2.9.1.2.3.5.2.2.7.9
MMULT 7.9.1.2.22.8.2.1.12.10
READY

OUTPUT AREA

REPLY AREA

Figure V-16

Third page of commands for scaling example

and has determinant .80847, compared with .80373 for the minimum-determinant solution.

This example was carried out on the IBM 2250, using conversational OMNITAB, by R. E. Bargmann, in 3 1/2 hours, at a time when the system had low priority and response was sluggish. Again, the most helpful feature was the facility to look at the worksheet after almost every single command.

D. MULTIVARIATE ANALYSIS

The following example, the calculation of canonical and canonical-partial correlation, was carried out by students at the University of Georgia, in the Multivariate Methods class (STA 825). They were encouraged to use the interactive OMNITAB system. In supervising the operation, the instructor (R.E. Bargmann) had to advise students on efficient techniques of organizing the worksheet in such a way that related intermediate results would appear in the same worksheet section. A particular execution of this problem is shown in the displays in this section. Comments reported to the author by several students have also been included. These comments serve to point out the variety of approaches available and to illustrate some of the most common difficulties encountered by students.

During the presentation of a course in multivariate methods, one of the important concepts is that of correlation between sets of

variables. Consequently, the instructor will introduce the concept of canonical correlation and derive step-by-step algorithms for the evaluation of relevant results. The population canonical correlation will be shown to be the square root of the largest characteristic root of

$$\Sigma_{11}^{-1} \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{12}' \quad \text{where } \Sigma_{11} = \text{Var}(\underline{x}), \Sigma_{22} = \text{Var}(\underline{y}) \text{ and } \Sigma_{12} = \text{COV}(\underline{x}, \underline{y}').$$

$\hat{\underline{a}}$, the vector of weights associated with \underline{x} , is the associated characteristic vector and $\hat{\underline{b}}$, the vector of weights associated with \underline{y} , is

$\Sigma_{22}^{-1} \Sigma_{12}' \hat{\underline{a}}$. The process of maximum-likelihood estimation is performed in the following steps:

- (1) Obtain R , the matrix of sample correlations.

$$R = \begin{bmatrix} R_{11} & R_{12} \\ R_{12}' & R_{22} \end{bmatrix}$$

- (2) Obtain R_{11}^{-1} and R_{22}^{-1} .

- (3) Obtain $R_{22}^{-1} R_{12}'$.

- (4) Obtain $R_{12} R_{22}^{-1} R_{12}'$.

- (5) Obtain $R_{11}^{-1} R_{12} R_{22}^{-1} R_{12}'$.

- (6) Obtain the largest characteristic root of the matrix obtained in (5).

- (7) Obtain $\hat{\rho}(\underline{x}, \underline{y})$, the square root of (6)

(8) Obtain $\hat{\underline{a}}$, an associated characteristic vector from (5) and (6).

(9) Obtain $\hat{\underline{b}} = R_{22}^{-1} R_{12}' \hat{\underline{a}}$.

With these steps in mind, a student would write the following OMNITAB statements (see Figure V-22) to obtain $\hat{\rho}(x,y)$. Step (1) requires him to obtain or enter the correlation matrix. In this example the correlation matrix of order 8 by 8 was entered, from the console, into the worksheet, starting in location (1,1) following the command `READ 1***8`. Thus the raw data would appear in two adjacent sections of the worksheet, Figures V-17 and V-18. It is useful to check input data before further calculations are performed. In this example, an error was found in row 7. To correct this error it was necessary only to type ROW 7 on the console (since control is still in the READ mode) and then to enter the corrected row as illustrated in Figure V-19. During discussions with students, it was discovered that they had some difficulty entering the data. One problem is caused, of course, by a lack of typing skill, which again shows that a minimum typing facility is necessary for all interactive work with a computer. Another common problem occurred when a student attempted to enter a row of data before the cue was given (ROW N); such data will be lost.

In step (2) we find R_{zz}^{-1} and R_{yy}^{-1} .

```
MINVERT RZZ 4,4 2,2 10,1
MINVERT RYY 1,1 3,3 13,1
```

OUTPUT AREA					
WORKSHEET PART 1					
C O L U M N S					
	1	2	3	4	5
1	1.00000	0.690000	0.596000	0.260000	0.100000
2	0.690000	1.00000	0.690000	0.260000	0.200000
3	0.596000	0.690000	1.00000	0.255000	0.146000
4	0.260000	0.285000	0.255000	1.00000	0.389000
5	0.100000	0.200000	0.146000	0.389000	1.00000
6	0.421000	0.397000	0.386000	0.321000	0.162000
7	0.510000	0.300000	0.252000	0.370000	0.230000
8	0.376000	0.349000	0.229000	0.408000	0.303000
9	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0
36	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0
REPLY AREA					

Figure V-17

Worksheet after entering correlation matrix

OUTPUT AREA					
WORKSHEET PART 2					
C O L U M N S					
	6	7	8	9	10
1	0.421000	0.353000	0.376000	0.0	0.0
2	0.397000	0.300000	0.349000	0.0	0.0
3	0.386000	0.252000	0.329000	0.0	0.0
4	0.321000	0.370000	0.408000	0.0	0.0
5	0.162000	0.236000	0.303000	0.0	0.0
6	1.000000	0.611000	0.642000	0.0	0.0
7	0.611000	0.642000	0.642000	0.0	0.0
8	0.642000	0.642000	1.000000	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0
36	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0
REPLY AREA					

Figure V-18

Worksheet after entering correlation matrix

OUTPUT AREA
 IF THE SCREEN BECOMES FULL AN ALARM WILL SOUND. WHEN YOU WANT TO SEE
 THE NEXT SECTION OF YOUR PROGRAM, PRESS KEY 2.

ERASE
 READ 1 *** 8
 1 .68 .596 .26 .165 .421 .35 .376
 .69 1 .655 .285 .2 .397 .3 .349
 .596 .655 1 .255 .146 .386 .252 .329
 .26 .285 .255 .398 .321 .37 .408
 .165 .2 .146 .398 1 .162 .236 .303
 .421 .397 .386 .321 .162 1 .611 .642
 .35 .3 .252 .37 .236 .611 .642
 .376 .349 .329 .408 .303 .642 .576 1
 ROW 7:
 .35 .3 .252 .37 .236 .611 1 .576
 ROW 8:

REPLY AREA

Figure V-19

Changing the row counter while entering data

MINVERT is the matrix inversion command. RZZ and RYY are comments inserted as aids in following the logic. Note that the lower right-hand partition of R is R_{ZZ} . This partition starts in coordinate (4,4). The first two arguments of the command indicate the upper left-hand corner of the matrix to be inverted; the third and fourth arguments contain the dimensions of the matrix and the last two arguments specify the upper left-hand corner of the resulting matrix.

Step (3) requires the calculation of $R_{yy}^{-1}R'_{zy}$. However, in this example this result is not necessary since only the canonical correlation is desired. Therefore, the student would proceed directly to step (4) in which he obtains $R_{zy}R_{yy}^{-1}R'_{zy}$. This result can be obtained in one statement by using a special matrix command as follows:

M(XAX') 13,1 3,3 4,1 2,3 17,1 .

This command can be used to obtain the matrix product XAX' . The first four arguments indicate the location and size of A, while arguments 5 through 8 indicate the location and size of X. The last two arguments are used to indicate the worksheet location of the result.

Step (5) is performed as a matrix multiplication, using the result in (4), by entering the command

MULT 10,1 2,2 17,1 2,2 20,1

where the first four arguments define the first matrix and the second four define the second matrix used in the matrix multiplication. As before the last two arguments indicate where the result is to be stored.

Figure V-20 shows the results of the preceding commands. At the time this illustration was run, eigenvalue and eigenvector routines had

OUTPUT AREA
WORKSHEET PART 1

C O L U M N S

	1	2	3	4	5
1	1.00000	0.690000	0.536000	0.200000	0.165000
2	0.690000	1.00000	0.655000	0.295000	0.200000
3	0.590000	0.655000	1.00000	0.255000	0.146000
4	0.260000	0.285000	0.255000	1.00000	0.398000
5	0.165000	0.200000	0.146000	0.398000	1.00000
6	0.421000	0.357000	0.390000	0.321000	0.162000
7	0.350000	0.300000	0.252000	0.370000	0.236000
8	0.376000	0.349000	0.329000	0.408000	0.303000
9	0.0	0.0	0.0	0.0	0.0
10	1.18822	-0.472911	0.0	0.0	0.0
11	-0.472911	1.19822	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	2.05104	-1.07628	-0.517452	0.0	0.0
14	-1.07628	2.31617	-0.875627	0.0	0.0
15	-0.517452	-0.875627	1.88194	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.937622E-01	0.609906E-01	0.0	0.0	0.0
18	0.609906E-01	0.414995E-01	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0
20	0.825607E-01	0.528446E-01	0.0	0.0	0.0
21	0.281290E-01	0.204673E-01	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0
36	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0

REPLY AREA

Figure V-20

Worksheet after entering REULT 10,1 2,2 17,1 2,2 20,1

not been incorporated, hence steps (6) and (7) had to be performed on a desk calculator, an easy operation in this instance.

The advantages of having an interactive worksheet system are even more pronounced in the next phase of this assignment following the introduction of the concept of canonical partial correlation. This concept requires that the instructor explain the conditional variance-covariance matrix through a discussion of conditional distributions. This conditional variance-covariance matrix can be represented in general as $\Sigma_{ij}^* = \Sigma_{ij} - \Sigma_{ik} \Sigma_{kk}^{-1} \Sigma_{jk}$, the covariance matrix for the i'th and j'th sets with the k'th set partialled out. To obtain a canonical partial correlation one needs only to replace the variance-covariance matrices used in steps (1) through (7) above by the conditional variance-covariance matrices. This procedure can be outlined in the following steps:

$$(1a) \text{ Calculate } R_{11}^* = R_{11} - R_{13} R_{33}^{-1} R_{13}.$$

$$(2a) \text{ Calculate } R_{22}^* = R_{22} - R_{23} R_{33}^{-1} R_{23}.$$

$$(3a) \text{ Calculate } R_{12}^* = R_{12} - R_{13} R_{33}^{-1} R_{23}.$$

(4a) Perform steps (1) through (7) given above.

These steps can be carried out using the following MONITAB procedure. In this example, students had to find the canonical correlation between sets y and z after partialing out set u. Thus, the first statement would be

MINVERT RUU 6,6 3,3 23,1

since R_{uu}^{-1} is needed to obtain the conditional variance-covariance matrices. This statement would be followed by

$N(XAX') \ 23,1 \ 3,3 \ 1,6 \ 3,3 \ 27,1$

to obtain $R_{yu} R_{uu}^{-1} R_{yu}'$.

At this point the 8 by 8 correlation matrix is moved into the lower half of the worksheet (MOVE 1,1 8-8 41,1) so that the subtractions can be carried out in the upper half. None of the students interviewed moved the correlation matrix as is done here. They all stored R_{11}^* , R_{22}^* and R_{12}^* wherever they could within the first section of the worksheet. All of the students also kept all results, both intermediate and final, within the first section of the worksheet, replacing earlier results. The transfer of the original correlation matrix to a new region, and the subsequent construction of the conditional covariance matrices into the field of the original correlation matrix, made the calculation of canonical-partial correlation identical to that of the canonical correlation. The students, in not following this procedure, were required to re-structure the coordinates of the matrices needed in intermediate calculations.

As a consequence of moving the correlation matrix, the canonical partial correlation can now be obtained by repeating the commands used earlier to obtain the canonical correlation. It may be of interest to note that the facility for repeatedly executing a sequence of commands is available in the batch version of OMNITAB. This feature was not incorporated within the interactive version because problems which require extensive repetition of a section of code should be performed in the batch mode.

Step (1a) is completed by entering the command

```
MSUB 1,1 3,3 27,1 3,3 1,1 .
```

Step (2a) requires the same sequence of commands as were required for step (1a) and thus can be entered very quickly as follows:

```
M(XAX') 23,1 3,3 4,6 2,3 31,1
MSUB 4,4 2,2 31,1 2,2 4,4 .
```

To obtain R_{zy}^* the command M(XAX') must be replaced by a pair of MMULT commands.

```
MMULT 23,1 2,3 6,1 3,3 34,1
MMULT 4,6 2,3 34,1 3,3 38,1
MSUB 4,1 2,3 38,1 2,3 4,1 .
```

To perform step (4a), it is necessary only to repeat the steps required earlier to obtain the canonical correlation.

```
MINVERT RZZSTAR 4,4 2,2 10,1
MINVERT RYYSTAR 1,1 3,3 13,1
M(XAX') 13,1 3,3 4,1 2,3 17,1
MMULT 10,1 2,2 17,1 2,2 20,1
```

These results can be seen in Figure V-21. From these results the canonical partial correlation can be easily obtained.

This example illustrates the advantage to be gained in certain classes through the use of OMNITAB. Figure V-22 shows a listing of the complete set of commands used in this example. The program was relatively easy to write and was written in a much shorter period of time than would have been required to obtain the same results using calculators or computer programs.

Figure V-23 shows the approach of a student who followed a slightly different procedure. His comment after the session was that, "unfamiliarity presented a slight problem at first but after executing several commands I became familiar with the operations and then had no

OUTPUT AREA
WORKSHEET PART 1

C O L U M N S

	1	2	3	4	5
1	0.797481	0.502997	0.419861	0.260000	0.165000
2	0.502997	0.825941	0.489212	0.260000	0.200000
3	0.419861	0.489212	0.839642	0.260000	0.140000
4	0.769256E-01	0.119782	0.105001	0.806053	0.263087
5	0.531605E-01	0.100141	0.571364E-01	0.783087	0.696558
6	0.421000	0.397000	0.396000	0.321000	0.162000
7	0.350000	0.300000	0.282000	0.370000	0.236000
8	0.376000	0.349000	0.329000	0.409000	0.303000
9	0.0	0.0	0.0	0.0	0.0
10	1.37202	-0.402608	0.0	0.0	0.0
11	-0.402608	1.23352	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	2.15462	-1.02917	-0.477773	0.0	0.0
14	-1.02917	2.34034	-0.848953	0.0	0.0
15	-0.477773	-0.848953	1.92453	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.195076E-01	0.144483E-01	0.0	0.0	0.0
18	0.144483E-01	0.122662E-01	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0
20	0.209479E-01	0.148849E-01	0.0	0.0	0.0
21	0.996829E-02	0.931361E-02	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0
23	1.99683	-0.720831	-0.966933	0.0	0.0
24	-0.720831	1.75670	-0.549087	0.0	0.0
25	-0.966833	-0.549086	1.87278	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0
27	0.202519	0.187003	0.176139	0.0	0.0
28	0.187003	0.174059	0.165788	0.0	0.0
29	0.176139	0.165788	0.150956	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0
31	0.195947	0.154913	0.0	0.0	0.0
32	0.154913	0.103442	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0
34	0.262489	0.274009	0.303979	0.0	0.0
35	0.104919	0.492058E-01	-0.160019E-01	0.0	0.0
36	0.147048	0.144741	0.143177	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0
38	0.183074	0.155218	0.143989	0.0	0.0
39	0.111839	0.99582E-01	0.99582E-01	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0

REPLY AREA

Figure V-21

Worksheet after entering MMULT 10,1 2,2 17,1 2,2 20,1

THIS IS FSTAR

OUTPUT AREA
 IF THE SCREEN BECOMES FULL AN ALARM WILL SOUND. WHEN YOU WANT TO SEE
 THE NEXT SECTION OF YOUR PROGRAM, PRESS KEY 2.

```

ERASE
READ 1 *** 8
1 .69 .596 .26 .165 .421 .35 .376
.03 1 .655 .205 .2 .397 .3 .349
.596 .655 1 .255 .146 .386 .252 .329
.26 .295 .355 1 .398 .321 .37 .408
.165 .2 .146 .398 1 .162 .236 .303
.421 .397 .386 .321 .162 1 .611 .642
.35 .3 .252 .37 .236 .611 .642
.376 .349 .329 .408 .303 .642 .576 1
RTN 7
.35 .3 .252 .37 .236 .611 1 .576
MINVERT RZZ 4.4 2.2 10.1
MINVERT RYY 1.1 3.3 13.1
MINXAX 13.1 3.3 4.1 2.3 17.1 $RZY=RYYINVERSE=RZYPRIME
MMULT 10.1 2.2 17.1 2.2 20.1
MINVERT RUU 6.6 3.3 23.1
MINXAX 23.1 3.3 1.6 3.3 27.1
MMOVE 1.1 8.8 40.11
MMOVE 1.1 8.8 41.11
MSUB 1.1 3.3 27.1 3.3 1.1 RYYSTAR
MINXAX 23.1 3.3 4.6 2.3 31.1
MSUB 4.4 2.2 31.1 2.2 4.4 RZZSTAR
MMULT 23.1 3.3 6.1 3.3 34.1
MMULT 4.6 2.3 34.1 3.3 38.1
MSUB 4.1 2.3 38.1 2.3 4.1
MINVERT RZZSTAR 4.4 2.2 10.1
MINVERT RYYSTAR 1.1 3.3 13.1
MINXAX 13.1 3.3 4.1 2.3 17.1
MMULT 10.1 2.2 17.1 2.2 20.1 THIS IS FSTAR
READY
  
```

REPLY AREA

Figure V-22
 Commands used for correlation example

```

MINVERT 4 4 2 2 10 1
MINVERT 1 1 3 3 13 1
M(XAX') 13 1 3 3 4 1 2 3 17 1
MMULT 10 1 22 17 1 2 2 20 1
MINVERT 6 6 3 3 23 1
M(XAX') 23 1 3 3 1 6 3 3 27 1
MMOVE 1 1 3 3 31 1
MSUB 31 1 3 3 27 1 3 3 31 1
M(XAX') 23 1 3 3 4 6 2 3 10 4
MSUB 4 4 2 2 10 4 2 2 13 4
MMULT 4 6 2 3 23 1 3 3 35 1
MTRANS 1 6 3 3 38 1
MMULT 35 1 2 3 38 1 3 3 35 1
MSUB 4 1 35 1 2 3 35 1
MINVERT 31 1 3 3 27 1
M(XAX') 27 1 3 3 35 1 2 3 17 3
MINVERT 13 4 2 2 20 4
MMULT 20 4 2 2 17 3 2 2 30 4

```

Figure V-23

A second approach to the correlation example

difficulty." Another student who had had some experience with the batch version at Iowa State, expressed a preference for the Interactive OMNITAB version because he "saw all results right away." Since this was the very purpose of the development of our interactive version, this comment seems to indicate that some measure of success has been achieved.

BIBLIOGRAPHY

- [1] Bargmann, Rolf E. "A Statistical Distribution Computer Package." Department of Statistics and Computer Science, University of Georgia.
- [2] Bargmann, R. E. "Principles and Designs of Statistical Computer Languages." Chapter 8 of Progress in Operations Research, Volume III, New York: John Wiley & Sons, Inc., (1969).
- [3] Barrodale, Ian. "Certification of Algorithm 127, ORTHO." Communications of the ACM, 13, p. 122, (1970).
- [4] Bartky, Ian R. "The $A^1\Sigma - 1\Sigma$ Transition of ^{39}KH and ^{39}KD . Vibrational Numbering and Molecular Constants." Journal of Molecular Spectroscopy, 20, pp. 299-311, (1966).
- [5] Beam, Alfred E., and Joseph Hilsenrath. PRECISE: A Multiple Precision Version of Omnitab. National Bureau of Standards, Technical Note 446, (1968).
- [6] Blackburn, G. F. and F. R. Caldwell. "Reference Tables for Thermo couples of Iridium - Rhodium Alloys Versus Iridium." Journal of Research of the National Bureau of Standards-C. Engineering and Instrumentation 68C, pp. 41-59, (1964).
- [7] Bourdeau-Martini, Jeannine and Carl R. Honig. "Control of Coronary Intercapillary Distance: Effect of Arterial PCO_2 and pH." Microvascular Research 6, pp. 286-296, (1973).
- [8] Bouver, Hubert. Curve Fitting by the Method of Moments. Technical Report #102, Department of Statistics and Computer Science, University of Georgia, (1973).
- [9] Braun, W. and T. Carrington. "Line Emission Sources For Concentration Measurements and Photochemistry." Journal of Quantitative Spectroscopy and Radiative Transfer 9, pp. 1133-1143, (1969).
- [10] Braun, W., Arnold M. Bass and D. D. Davis. "Experimental Test of a Two-Layer Model Characterizing Emission-Line Profiles." Journal of the Optical Society of America 60, pp. 166-170, (1970).

- [11] Braun, W., Arnold M. Bass and M. Pilling. "Flash Photolysis of Ketene and Diazomethane: The Production and Reaction Kinetics of Triplet and Singlet Methylene." The Journal of Chemical Physics 52, pp. 5131-5143, (1970).
- [12] Brooks, C., S. R. Hart and I. Wendt. "Realistic Use of Two-Error Regression Treatments as Applied to Rubidium-Strontium Data." Reviews of Geophysics and Space Physics 10, pp. 551-577, (1972).
- [13] Cady, F. B. and W. A. Fuller. "The Statistics-Computer Interface in Agronomic Research." Agronomy Journal 62, pp. 599-604, (1970).
- [14] Carnahan, B., H. Luther and J. Wilkes. Applied Numerical Analysis. New York: John Wiley & Sons, Inc., (1969).
- [15] Chamberlain, R. L. OMNITAB: The Operating System of the FORTRAN Version. Statistical Laboratory, Iowa State University, (1968).
- [16] Chang, Jeffrey Chit-Fu and Rolf E. Bargmann. Internal Multi-dimensional Scaling of Categorical Variables. Technical Report #108, Department of Statistics and Computer Science, University of Georgia, (1974).
- [17] Criswell, B. Sue, William T. Butler, Roger D. Rossen and Vernon Knight. "Marine Malaria The Role of Humeral Factors and Microphages in Destruction of Parasitized Erythrocytes." Journal of Immunology 107, pp. 212-221, (1971).
- [18] Davies, Helen W. "Calibration of the Nickel Dimethylglyoxine Spectral Shift at Pressures to 20 Kilobars for Use in Spectroscopic Pressure Measurement." Journal of Research of the National Bureau of Standards - A. Physics and Chemistry 72A, pp. 149-153, (1968).
- [19] Davis, Philip and Philip Rabinowitz. "A Multiple Purpose Orthonormalizing Code." Journal of the Association for Computing Machinery 1, pp. 183-191, (1954).
- [20] Davis, Philip J. and Philip Rabinowitz. "Advances in Orthonormalizing Computation." Advances in Computers 2, pp. 55-133, (1961).
- [21] Davis, Philip J. "Orthonormalizing Codes in Numerical Analysis." Chapter 10 of Survey of Numerical Analysis. New York: McGraw-Hill, (1962).
- [22] Dickson, R. W., J. B. Wachtman, Jr., and S. M. Copley. "Elastic Constants of Single-Crystal Ni₃Al from 10° to 850° C." Journal of Applied Physics 40, pp. 2276-2279, (1969).

- [23] Edwards, J. L. and D. P. Johnson. "A Dynamic Method for Determining the Vapor Pressure of Carbon Dioxide at 0° C." Journal of Research of the National Bureau of Standards - C. Engineering and Instrumentation 72C, pp. 27-32, (1968).
- [24] Evans, J. P. and S. D. Woods. "An Intercomparison of High Temperature Platinum Resistance Thermometers and Standard Thermocouples." Metrologia 7, pp. 108-130, (1971).
- [25] Falkoff, A. D. and K. E. Iverson. "APL/360 Terminal System." Proceedings of the Symposium on Interactive Systems for Experimental Applied Mathematics. New York: Academic Press, Inc., (1968).
- [26] Fisher, R. A. "The Precision of Discriminant Functions." Annals of Eugenics, London 10, pp. 422-429, (1940).
- [27] Fletcher, R., and M.J.D. Powell. "A Rapidly Convergent Decent Method for Minimization." Computer Journal 6, pp. 163-168, (1963).
- [28] Hahn, Thomas A. "Thermal Expansion of Copper from 20 to 800 K - Standard Reference Material 736." Journal of Applied Physics 41, pp. 5096-5101, (1970).
- [29] Harmon, H. H. Modern Factor Analysis. Chicago: University of Chicago Press, (1960).
- [30] Heydemann, P. L. M. and J. C. Houck. "Bulk Modulus and Density of Polyethylene to 30 Kbar." Journal of Polymer Science: Part A-2 10, pp. 1631-1637, (1972).
- [31] Hilsenrath, Joseph; Guy G. Ziegler; Carla G. Messina; Philip J. Walsh; and Robert J. Herbold. OMNITAB: A Computer Program for Statistical and Numerical Analysis. National Bureau of Standards, Handbook 101, (1968).
- [32] Hogben, David; Sally T. Peavy; and Ruth N. Varner. OMNITAB II: User's Reference Manual. National Bureau of Standards, Technical Note 552, (1971).
- [33] Horton, William S. "Statistical Aspects of Second and Third Law Heats." Journal of Research of the National Bureau of Standards - A. Physics and Chemistry 70A, pp. 533-539, (1966).
- [34] Horton, William S. "Anisotropic Reaction Kinetics of Oxygen with Pyrolytic Graphite." Journal of Research of the National Bureau of Standards - A. Physics and Chemistry 74A, pp. 325-330, (1970).

- [35] Hyland, Richard W. and Arnold Wexler. "The Enhancement of Water Vapor in Carbon Dioxide - Free Air at 30, 40, and 50° C." Journal of Research of the National Bureau of Standards - A. Physics and Chemistry 77A, pp. 115-131, (1973).
- [36] IMSL Numerical Computations Newsletter 1, (1972)
- [37] Jowett, D., and R. L. Chamberlain. The OMNITAB Programming System: A Guide for Users. Houston, Texas: Shell Oil Company, (1968).
- [38] Jowett, D.; R. L. Chamberlain; and A. G. Mexas. "OMNITAB - A Single Language for Statistical Computations." Journal of Statistical Computation and Simulation 1, pp. 129-147, (1972).
- [39] Lafferty, Walter J. and Robert J. Thibault. "High-Resolution Infrared Spectra of $\text{Cl}_2^{12}\text{H}_2$, $\text{Cl}_2^{12}\text{C}^{13}\text{H}_2$, and C_2H_2 ." Journal of Molecular Spectroscopy 14, pp. 79-96, (1964).
- [40] Lafferty, Walter J., Arthur G. Maki and Earle K. Plyler. "High-Resolution Infrared Determination of the Structure of Carbon Suboxide." Journal of Chemical Physics 40, pp. 224-229, (1964).
- [41] Lancaster, H. P. "Some Properties of the Bivariate Normal Distribution in the Form of a Contingency Table." Biometrika 44, pp. 289-292, (1957).
- [42] Lancaster, H. P. "The Structure of Bivariate Distributions." Annals of Mathematical Statistics 29, pp. 719-736, (1958).
- [43] Longley, James W. "An Appraisal of Least Squares Programs for the Electronic Computer from the Point of View of the User." Journal of the American Statistical Association 62, pp. 819-841, (1967).
- [44] Maki, Arthur G. "Measurement of the Direct 1-Douplet Transitions in Carbonyl Sulfide." Journal of Molecular Spectroscopy 23, pp. 110-111, (1967).
- [45] Maki, Arthur G., Jr. and David R. Lide, Jr. "Microwave and Infrared Measurements on HCN and DCN: Observations on 1-Type Resonance Doublets." The Journal of Chemical Physics 47, pp. 3206-3210, (1967).
- [46] Muller, Mervin E. "Computers as an Instrument for Data Analysis." Technometrics 12, pp. 259-293, (1970).
- [47] Penn, Lu. An On-Line Statistical Computer System for Lay Usage. Department of Statistics, University of Georgia, (1971).

- [48] Robbins, C. R. and E. M. Levin. "Phase Transformation in Barium Tetraborate." Journal of Research of the National Bureau of Standards - A. Physics and Chemistry 73A, pp. 615-620, (1969).
- [49] Rosenblatt, Joan R., Brian L. Joiner, and David Hogben. "OMNITAB-Rapid Statistical Manipulation." Final 1970 Census Plans and Four Programming Systems for Computerized Data Retrieval and Manipulation. Census Tract Papers, Series GE-40, No. 6 Bureau of the Census, (1969).
- [50] Ryan, T. A., Jr., and B. L. Joiner. MINITAB Commands, Statistics Department, Pennsylvania State University, (1972).
- [51] Ryan, T. A., Jr. and Brian L. Joiner. "Minitab: A Statistical Computing System for Students and Researchers." The American Statistician 27, pp. 222-225, (1973).
- [52] Ryan, T. A., Jr. and B. L. Joiner. MINITAB: Student Statistical Program. Statistics Department, Pennsylvania State University.
- [53] Samnet, Jean E. Programming Languages: History and Fundamentals. Englewood Cliffs, N.J.: Prentice-Hall, (1969).
- [54] Shimanouchi, Takehiko and Isao Suzuki. "Method of Adjusting Force Constants and its Application to H_2O , H_2CO , CH_2Cl and Their Deuterated Molecules." Journal of Chemical Physics 42, pp. 296-308, (1965).
- [55] Siegel, Sidney. Nonparametric Statistics for the Behavioral Sciences. New York: McGraw-Hill, (1956).
- [56] Steel, Robert G. D., "Minimum Generalized Variance for a Set of Linear Functions" Annals of Mathematical Statistics 22, pp. 456-460, (1951).
- [57] Story, V. M., D. McIntyre and J. H. O'Mara, "Solvent Effects on the Ultraviolet Absorption of Polystyrene." Journal of Research of the National Bureau of Standards - A. Physics and Chemistry 71 A, pp. 169-175, (1967).
- [58] Swanson, James M., Alexa Ledlow and Scott Harris. "Using OMNITAB Interactively in a Statistics Laboratory." Behavior Research Methods and Instrumentation 5, pp. 199-204, (1973).
- [59] Utton, D. B. "Temperature Dependence of the Nuclear Quadrupole Resonance Frequency of ^{35}Cl in $KClO_3$ between 120 and 900 K." The Journal of Chemical Physics 47, pp. 371-373, (1967).

- [60] Vincentini-Missoni, M., J.M.H. Levelt Sengers and M.S. Green. "Scaling Analysis of Thermodynamics Properties in the Critical Region of Fluids." Journal of Research of the National Bureau of Standards - A. Physics and Chemistry 73A, pp. 563-583, (1969).
- [61] Walsh, Philip J. "Algorithm 127. ORTHO." Communications of the ACM 5, pp. 511-513, (1962).
- [62] Wampler, Roy H., "An Evaluation of Linear Least Squares Computer Programs." Journal of Research of The National Bureau of Standards - B. Mathematical Sciences 73B, pp. 59-90, (1969).
- [63] Wampler, R. H. "A Report on the Accuracy of Some Widely Used Least Squares Computer Programs!" Journal of the American Statistical Association 65, pp. 549-565, (1970).

APPENDIX

PROGRAM LISTINGS

	SUBROUTINE AARGS	AARG	1
	COMMON / BLOCKA/MODE,H,KARD(77),AARG,ARG,ARG2,NEHCD(19),KRDEND	AARG	2
	1,NEHCOS(19,5),KSAVE,KSAVE,NFLAD	AARG	3
C		AARG	4
C	THIS SUBROUTINE ASSEMBLES A NUMBER FROM A STRING OF DIGITS.	AARG	5
C	H INITIALLY POINTS AT THE FIRST DIGIT. IT IS LEFT POINTING AT THE	AARG	6
C	FIRST CHARACTER AFTER THE NUMBER. THE VALUE OF THE NUMBER IS	AARG	7
C	RETURNED IN ARG. KARG IS USED TO INDICATE THE TYPE OF NUMBER.	AARG	8
C	KARG = 1 - FLOATING POINT	AARG	9
C	KARG = 0 - INTEGER	AARG	10
C	KARG = -1 - ERROR	AARG	11
	ARG=KARD(H)	AARG	12
	SIO = 1.	AARG	13
	JEXP=0	AARG	14
	IXS=1	AARG	15
	JEXP=0	AARG	16
	KARG=0	AARG	17
	KEXP = 1	AARG	18
C	LOOK BACK FOR MINUS SIGN AND/OR DECIMAL POINT	AARG	19
C		AARG	20
	K=KARD(H-1)	AARG	21
	IF(K.NE.37)GO TO 10	AARG	22
	KARG=1	AARG	23
	JEXP=-1	AARG	24
	K=KARD(H-2)	AARG	25
10	IF(K.EQ.38)SIO = -1.	AARG	26
20	H=H-1	AARG	27
	K=KARD(H)	AARG	28
	IF(K.EQ.10)GO TO 30	AARG	29
	KEXP=KEXP+1	AARG	30
	JEXP=JEXP-KARG	AARG	31
	ARG=10.*ARG+FLOAT(K)	AARG	32
	GO TO 20	AARG	33
30	IF(K.NE.37)GO TO 50	AARG	34
C	DECIMAL POINT FOUND	AARG	35
C		AARG	36
	IF(KARG.EQ.0)GO TO 40	AARG	37
	CALL ERROR(3)	AARG	38
	KARG=-1	AARG	39
	RETURN	AARG	40
40	KARG=1	AARG	41
	GO TO 27	AARG	42
C		AARG	43
C	CHECK FOR EXPONENT E X, E+X, E-X, +X, -X	AARG	44
C		AARG	45
50	IF(K.NE.14)GO TO 54	AARG	46
	H = H + 1	AARG	47
	K = KARD(H)	AARG	48
	IF(K.NE.44) IF(K = 10) 58, 54, 54	AARG	49
52	H = H + 1	AARG	50
	K = KARD(H)	AARG	51
	IF(K = 10) 50, 107, 100	AARG	52
54	IF(K.NE.30) IF(K = 39) 100, 50, 100	AARG	53
	IXS = -1	AARG	54

```

      00 TO 62
56  KARD=1
70  JEXP=10+JEXP+K
      M=M+1
      K=KARD(M)
      IF( K, .LT. 10 ) 00 TO 70
C
C      DONE WITH ARGUMENT
C
100  IF(KARD.NE.0)00 TO 120
      IF(KEXP.LE.10) 00 TO 110
      KARD=1
110  ARD=610 +ARD
      RETURN
120  IEXP = IX6 + JEXP + IEXP
      JEXP = JARD( IEXP )
      IF(ABS(IEXP+KEXP).GT.74) 00 TO 130
      IF( IEXP ) 123, 110, 120
123  ARD = ARD / 10. ** JEXP
      00 TO 110
120  ARD = ARD * 10. ** JEXP
      00 TO 110
130  CALL ERROR( 102 )
      ARD = 0.
      00 TO 110
      END

```

```

ARD 55
ARD 56
ARD 57
ARD 58
ARD 59
ARD 60
ARD 61
ARD 62
ARD 63
ARD 64
ARD 65
ARD 66
ARD 67
ARD 68
ARD 69
ARD 70
ARD 71
ARD 72
ARD 73
ARD 74
ARD 75
ARD 76
ARD 77
ARD 78
ARD 79
ARD 80

```



```

      GO TO 409
405 RC(1)=0.
      CALL ERROR(105)
406 I1 = I1 + KK( 1 )
410 I2 = I2 + KK( 2 )
      GO TO 10
500 DO 510 I = 13, JJ
      RC( I ) = FEXP2( RC( I1 ), RC( I2 ) )
      I1 = I1 + KK( 1 )
510 I2 = I2 + KK( 2 )
      GO TO 10
600 DO 610 I = 14, JJ
      RC( I ) = RC( I ) + ( RC( I1 ) + RC( I2 ) ) = RC( I3 )
      I1 = I1 + KK( 1 )
      I2 = I2 + KK( 2 )
610 I3 = I3 + KK( 3 )
      GO TO 10
700 DO 710 I = 14, JJ
      RC( I ) = RC( I ) + ( RC( I1 ) - RC( I2 ) ) = RC( I3 )
      I1 = I1 + KK( 1 )
      I2 = I2 + KK( 2 )
710 I3 = I3 + KK( 3 )
      GO TO 10
800 DO 810 I = 14, JJ
      RC( I ) = RC( I ) + ( RC( I1 ) = RC( I2 ) ) = RC( I3 )
      I1 = I1 + KK( 1 )
      I2 = I2 + KK( 2 )
810 I3 = I3 + KK( 3 )
      GO TO 10
900 DO 910 I = 14, JJ
      RC( I ) = RC( I ) + ( RC( I1 ) / RC( I2 ) ) = RC( I3 )
      I1 = I1 + KK( 1 )
      I2 = I2 + KK( 2 )
910 I3 = I3 + KK( 3 )
      GO TO 10
1000 DO 1010 I = 14, JJ
      RC( I ) = RC( I ) + RC( I3 ) = FEXP2( RC( I1 ), RC( I2 ) )
      I1 = I1 + KK( 1 )
      I2 = I2 + KK( 2 )
1010 I3 = I3 + KK( 3 )
      GO TO 10
      END

```

```

ARIT 55
ARIT 56
ARIT 57
ARIT 58
ARIT 59
ARIT 60
ARIT 61
ARIT 62
ARIT 63
ARIT 64
ARIT 65
ARIT 66
ARIT 67
ARIT 68
ARIT 69
ARIT 70
ARIT 71
ARIT 72
ARIT 73
ARIT 74
ARIT 75
ARIT 76
ARIT 77
ARIT 78
ARIT 79
ARIT 80
ARIT 81
ARIT 82
ARIT 83
ARIT 84
ARIT 85
ARIT 86
ARIT 87
ARIT 88
ARIT 89
ARIT 90
ARIT 91
ARIT 92
ARIT 93
ARIT 94
ARIT 95
ARIT 96

```

SUBROUTINE ARYVEC	ARYV 1
COMMON / BLOCKN/MODE,N,KARD(77),KARG,ARG,ARG2,NEHCO(19),KRCNO	ARYV 2
1,NEHCO(19,5),KSAVE,NSAVE,NFLAO	ARYV 3
COMMON / BLOCKN / RC(2433),IARGOS(5),KIND(39),ARGTAB(51),NRMAX,	ARYV 4
1 NR0H,NCJL,NHROS,VWXYZ(5)	ARYV 5
COMMON/BLOCKE/NAME(4),L1,L2,ISRFL0	ARYV 6
COMMON/SCRAT/N(00)	ARYV 7
C *****	ARYV 8
C THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS	ARYV 9
C H(AV) - L2=6	ARYV 10
C H(V'A) - L2=7	ARYV 11
C *****	ARYV 12
C CHECK FOR CORRECT NUMBER OF ARGUMENTS	ARYV 13
C *****	ARYV 14
IF(NHROS.NE.6.AND.NHROS.NE.7) GO TO 10	ARYV 15
C *****	ARYV 16
C CHECK TO SEE IF ALL ARGUMENTS ARE INTEGERS	ARYV 17
C *****	ARYV 18
J=NHROS	ARYV 19
CALL CRIND(J)	ARYV 20
IF(J.NE.0) GO TO 470	ARYV 21
C *****	ARYV 22
C COMPUTE ADDRESSES OF COLUMNS AND	ARYV 23
C CHECK TO SEE IF DIMENSIONS ARE OUT OF RANGE	ARYV 24
C *****	ARYV 25
CALL ADDRESS(5,19P)	ARYV 26
IF(IOP.EQ.0) GO TO 11	ARYV 27
IF(NHROS.EQ.7) GO TO 450	ARYV 28
J=1	ARYV 29
IF(L2.EQ.6) GO TO 440	ARYV 30
ICG=IARGOS(1)	ARYV 31
IF(ICG.LT.1.OR.ICG.GT.80) GO TO 470	ARYV 32
GO TO 400	ARYV 33
440 CALL ADDRESS(0,ICG)	ARYV 34
IF(ICG.EQ.0) GO TO 11	ARYV 35
GO TO 160	ARYV 36
450 IARGO(0)=IARGOS(0)	ARYV 37
IARGO(1)=IARGOS(1)	ARYV 38
IARGO(2)=1	ARYV 39
IARGO(3)=1	ARYV 40
IARGO(L2+1)=IARGO(L2-3)	ARYV 41
J=2	ARYV 42
400 CALL ATXCHK(J)	ARYV 43
IF(J-1) 490,470,400	ARYV 44
470 CALL ERSON(3)	ARYV 45
RETURN	ARYV 46
480 CALL ERSON (17)	ARYV 47
RETURN	ARYV 48
400 CALL PLUK	ARYV 49
IF(NFLAO.EQ.1) RETURN	ARYV 50
IAP=IARGO(1)	ARYV 51
IF(NHROS.EQ.7) ICG=IARGOS(5)	ARYV 52
IP=IARGO(L2-3)	ARYV 53
IF(L2.EQ.7) GO TO 640	ARYV 54

```

      JP=IARGC(4)
      IAD1=NRON
      IAD2=1
      GO TO 660
040  IAD1=1
      IAD2=PRON
      JP=IARGC(3)
060  DO 740 I=1,IP
      IA=IAP
      ID=IDP
      RI(I)=0.
      DO 680 J=1,JP
      RI(I)=RC(IA)*RC(ID)+R(I)
      IA=IA+IAD1
      ID=ID+1
      IAP=IAP+IAD2
      740 CONTINUE
C *****
C      STORE RESULTS IN WORKSHEET
C *****
      DO 800 I=1,IP
      RC(ICS)=R(I)
      ICS=ICS+IAD2
080  CONTINUE
      RETURN
10  CALL ERROR(10)
      RETURN
11  CALL ERROR(11)
      RETURN
      END

```

```

ARYV 55
ARYV 56
ARYV 57
ARYV 58
ARYV 59
ARYV 60
ARYV 61
ARYV 62
ARYV 63
ARYV 64
ARYV 65
ARYV 66
ARYV 67
ARYV 68
ARYV 69
ARYV 70
ARYV 71
ARYV 72
ARYV 73
ARYV 74
ARYV 75
ARYV 76
ARYV 77
ARYV 78
ARYV 79
ARYV 80
ARYV 81
ARYV 82
ARYV 83
ARYV 84

```

	SUBROUTINE ASTER	ASTE 1
	COMMON / BLOCKA/MODE,H,KARD(77),KARG,ARG,ARG2,NEUCD(19),KRDEND	ASTE 2
	1,NEUCD3(19,5),KSAVE,NSAVE,NFLAG	ASTE 3
	DIMENSION NAM(2)	ASTE 4
	THIS SUBROUTINE IS CALLED WHEN ASTERISKS ARE FOUND IN THE INPUT	ASTE 5
	LINE. KARG IS USED AS A CODE FOR BOTH INPUT AND OUTPUT.	ASTE 6
	AS INPUT KARG=1 - SINGLE ASTERISK	ASTE 7
	KARG=0 - DOUBLE ASTERISKS.	ASTE 8
	AS OUTPUT KARG=1 - ERROR	ASTE 9
	KARG=2 - FLOATING POINT CONSTANT	ASTE 10
	KARG=3 - INTEGER VARIABLE	ASTE 11
	KARG=4 - FLOATING POINT VARIABLE	ASTE 12
	KARG=5 - WORKSHEET ENTRY TO BE USED AS AN INTEGER	ASTE 13
	KARG=6 - WORKSHEET ENTRY TO BE USED AS A FLOATING	ASTE 14
	POINT NUMBER	ASTE 15
	KARG=7 - ASTERISKS INDICATING THROUGH	ASTE 16
		ASTE 17
	L=KARG	ASTE 18
	M=NONBLA(1)	ASTE 19
10	IF(K.NE.40)GO TO 20	ASTE 20
		ASTE 21
	A LONG LINE OF ASTERISKS FOUND	ASTE 22
		ASTE 23
	KARG=7	ASTE 24
15	M=M+1	ASTE 25
	IF(KARD(M).EQ.40) GO TO 15	ASTE 26
	GO TO 120	ASTE 27
20	IF(K.GE.36)GO TO 999	ASTE 28
	IF(K.GE.10)GO TO 50	ASTE 29
		ASTE 30
	NUMBER IS FIRST NON-BLANK CHARACTER. SET M = COMMA	ASTE 31
	M=43	ASTE 32
30	CALL ARRG3	ASTE 33
	IF(KARG.NE.0) GO TO 999	ASTE 34
	IF(NONBLA(1).EQ.M)IF(M-40)40,45,40	ASTE 35
	GO TO 999	ASTE 36
40	M = M + 1	ASTE 37
	IF(NONBLA(1).GE.10)GO TO 999	ASTE 38
		ASTE 39
	GET M = ASTERISK	ASTE 40
		ASTE 41
	M=40	ASTE 42
	Y=ARG	ASTE 43
	GO TO 30	ASTE 44
45	ARG2=ARG	ASTE 45
	ARG=Y	ASTE 46
	KARG=5	ASTE 47
	GO TO 100	ASTE 48
		ASTE 49
	LETTER FOUND FIRST	ASTE 50
		ASTE 51
50	CALL NAME(NAM(1))	ASTE 52
	CALL PHYCON(MD(1))	ASTE 53
		ASTE 54

	IF(IARO.EQ.0.)GO TO 00	ASTE 55
C		ASTE 56
C	PHYSICAL CONSTANT FOUND, SET KARO = 1	ASTE 57
C		ASTE 58
	KARO=1	ASTE 59
	IF(L.EQ.1) GO TO 00	ASTE 60
	GO TO 099	ASTE 61
C		ASTE 62
C	NAME NOT IN PHYSICAL CONSTANT LIST, TRY VARIABLE LIST	ASTE 63
C		ASTE 64
00	CALL VARCON(NAME(I))	ASTE 65
	IF(IARO.NE.0.)GO TO 80	ASTE 66
	CALL ERROR(0)	ASTE 67
70	KARO=1	ASTE 68
	RETURN	ASTE 69
00	KARO=0	ASTE 70
00	IF(NONDLA(I).NE.40)GO TO 090	ASTE 71
100	N=N+1	ASTE 72
C		ASTE 73
C	CHECK THAT THE NUMBER OF ASTERISKS AT THE END OF THE EXPRESSION	ASTE 74
C	IS THE SAME AS AT THE BEGINNING. L=0 MEANS 1, L=1 MEANS 2	ASTE 75
C		ASTE 76
	IF(L.NE.0)IF(KARO(N)-40)110,090,110	ASTE 77
	IF(KARO(N).NE.40.OR.KARO(N+1).EQ.40)GO TO 090	ASTE 78
	N=N+1	ASTE 79
110	KARO=KARO+L	ASTE 80
120	RETURN	ASTE 81
090	CALL ERROR(7)	ASTE 82
	GO TO 70	ASTE 83
	END	ASTE 84

BLOCK DATA	BLOC	1
COMMON / BLOCKN / NCTOP	BLOC	2
COMMON / BLOCKN/MODE,M,KARD(77),KARG,ARG,ARG2,NEWCO(19),KROEND	BLOC	3
1,NEWCOS(19,5),KSAVE,NSHVE,NFLAG	BLOC	4
COMMON/PCONST/P(2),N(2)	BLOC	5
COMMON / BLOCKN / RC(2499),IARGS(69),KIND(39),AROTAD(51),NRMAX,	BLOC	6
1 NRON,NCOL,NAROS,VWXYZ(5)	BLOC	7
COMMON/BLOCKE/NAME(4),L1,L2,ISRFLG	BLOC	8
COMMON/KPLOT/NFRAME,KKND,SIZE,SPACE	BLOC	9
COMMON/CONSTS/PI,E,HALFPI,DEG,RAD	BLOC	10
	BLOC	11
THIS BLOCK DEFINES CERTAIN CONSTANTS WHICH SERVE DIFFERENT	BLOC	12
FUNCTIONS IN THIS SYSTEM.	BLOC	13
	BLOC	14
DATA PI,E,HALFPI,DEG,RAD/3.141593,2.718282,	BLOC	15
1 1.570796,0.01745329,57.29578/	BLOC	16
DATA P,N/3.141593,2.718282,11907.3845/	BLOC	17
DATA MODE,KROEND,KSAVE,NSHVE,NFLAG/1,74,0.26,0/.NCTOP/1/.L1/0/	BLOC	18
DATA NRMAX,NRON,NCOL/0.00,30/.NFRAME,KKND,SIZE,SPACE/0.1,5,8,5,0/	BLOC	19
END	BLOC	20
SUBROUTINE CHANGE	CHAN	1
COMMON / BLOCKN/MODE,M,KARD(77),KARG,ARG,ARG2,NEWCO(19),KROEND	CHAN	2
1,NEWCOS(19,5),KSAVE,NSHVE,NFLAG	CHAN	3
COMMON / BLOCKN / RC(2499),IARGS(69),KIND(39),AROTAD(51),NRMAX,	CHAN	4
1 NRON,NCOL,NAROS,VWXYZ(5)	CHAN	5
	CHAN	6
THIS SUBROUTINE CHANGES SIGNS OF ELEMENTS IN COLUMNS SPECIFIED BY	CHAN	7
THE COMMAND CHANGE.	CHAN	8
	CHAN	9
IF(NAROS.LT.1) GO TO 010	CHAN	10
CALL CHNCOL(1)	CHAN	11
IF(1.EQ.1) GO TO 003	CHAN	12
IF(NRMAX.LT.1) GO TO 000	CHAN	13
CALL PLOK	CHAN	14
IF(NFLAG.EQ.1) RETURN	CHAN	15
DO 20 I=1,NAROS	CHAN	16
J=IAROS(1)	CHAN	17
DO 20 N=1,NRMAX	CHAN	18
JJ=J+N-1	CHAN	19
20 RC(JJ)=-RC(JJ)	CHAN	20
GO TO 000	CHAN	21
003 CALL ERROR (3)	CHAN	22
GO TO 007	CHAN	23
010 CALL ERROR (10)	CHAN	24
GO TO 003	CHAN	25
009 CALL ERROR(8)	CHAN	26
009 RETURN	CHAN	27
END	CHAN	28

	SUBROUTINE CHKCOL(J)	CHKC	1
	COMMON / BLOCKD / RC(2499),IAROS(69),KIND(39),AROTAB(51),NRMAX,	CHKC	2
	I NRON,NCOL,NAROS,VNXYZ(5)	CHKC	3
C		CHKC	4
C	THIS SUBROUTINE CHECKS TO SEE THAT IAROS(1) THROUGH IAROS(NAROS)	CHKC	5
C	ARE LEGAL COLUMN NUMBERS. RETURNS J=0 FOR NO ERROR AND	CHKC	6
C	J=1 FOR ERROR. IT ALSO CONVERTS IAROS(1) THROUGH IAROS(NAROS) TO	CHKC	7
C	THE BEGINNING ADDRESSES OF THE SPECIFIED COLUMNS IN THE WORKSHEET.	CHKC	8
C		CHKC	9
	IF(NAROS .GT. 0) GO TO 20	CHKC	10
10	J = 1	CHKC	11
	GO TO 40	CHKC	12
20	DO 30 I = 1, NAROS	CHKC	13
	CALL NORESS(I, IAROS(I))	CHKC	14
	IF(IAROS(I) .LE. 0) GO TO 10	CHKC	15
30	CONTINUE	CHKC	16
	J = 0	CHKC	17
40	RETURN	CHKC	18
	END	CHKC	19
	SUBROUTINE CKIND(J)	CKIN	1
	COMMON / BLOCKD / RC(2499),IAROS(69),KIND(39),AROTAB(51),NRMAX,	CKIN	2
	I NRON,NCOL,NAROS,VNXYZ(5)	CKIN	3
C		CKIN	4
C	THIS SUBROUTINE CHECKS THE FIRST J ARGUMENTS. IT RETURNS J=0 IF	CKIN	5
C	ALL ARE INTEGERS. J=1 IF ALL ARE REAL AND J=2 IF BOTH TYPES ARE	CKIN	6
C	FOUND.	CKIN	7
		CKIN	8
	JA=J	CKIN	9
	J=0	CKIN	10
	DO 10 I=1,JA	CKIN	11
	IF(KIND(I).NE.0) GO TO 15	CKIN	12
10	CONTINUE	CKIN	13
	RETURN	CKIN	14
15	J=1	CKIN	15
	DO 20 I=1,JA	CKIN	16
	IF(KIND(I).NE.1) GO TO 30	CKIN	17
20	CONTINUE	CKIN	18
	RETURN	CKIN	19
30	J=2	CKIN	20
	RETURN	CKIN	21
	END	CKIN	22

```

SUBROUTINE COMAND(M)                                COMA 1
DIMENSION NAMES(202),NWORK(14),M(7),TEXT(10),N1(100),N2(140) COMA 2
EQUIVALENCE (N1(1),NAMES(1)),(N2(1),NAMES(137))    COMA 3

THIS SUBROUTINE CAUSES A LIST OF AVAILABLE COMMANDS TO COMA 4
BE DISPLAYED ON THE 2260 SCREEN.                    COMA 5
                                                    COMA 6
                                                    COMA 7
                                                    COMA 8
                                                    COMA 9
DATA N1/'ARAD',' ','AARV','EC','ABS',' ','ABSO','LUTE', COMA 10
1'ACOS',' ','ACOS', COMA 11
2' 'O','ACOS','H','ACOT',' ','ACOT','O','ACOT','H','ADD',' ','ADEFCOMA 12
2'INE','ADIA','O','ADIV','IDE','AERA','SE','AROV','E','AMUL','T',COMA 13
3'ANTI','LOO','ARAI','SE','ARCC','OS','ARCC','OT','ARCS','IN','ARCTOMA 14
4'AN','ASCH','LAR','ASIN',' ','ASIN','O','ASIN','H','ASUS',' ','ACOMA 15
5TAN',' ','ATAN','O','ATAN','H','ATRA','NS','AVEC','ARR','AVEC','OICOMA 16
6AS','AVER','ACE','AZER','O','BETA','P','BETA','X','BETA','Z','DILOCCOMA 17
7'ATRA','CHAN','OE','CHIP',' ','CHIX',' ','CHIZ',' ','CLOS','E',COMA 18
8COS',' ','COSD',' ','COSH',' ','COT',' ','COTC',' ','COTH',' ','COCOMA 19
9UN','T','DEFI','NE','DEKO','TE','DEVN','OR','DIV',' ','DIVI','DE',COMA 20
A'DOPL','ICAT','ERAS','E','EXCH','ANCE','EXP',' ','EXPA','NO','EXPOCOMA 21
B'NENT','FFP',' ','FFX',' ','FFZ',' ','FLIP',' ','GAMP',' '/' COMA 22
DATA N2/'DANX', COMA 23
C' ','GANE',' ','GENE','RATE','HIER','ARCH','INVE','RT','LINE','ACOMA 24
DR','LOO',' ','LOOK',' ','LOOT','EN','HEAD',' ','HIAV',' ','HICA',COMA 25
E'','HIV',' ','HIX',' ','HIX','HIAA','X',' ','HIX',' ','HIXX',COMA 26
F'','HADD',' ','HAX',' ','HAXI','HUN','HDEF','INE','HDI','O','HGCCOMA 27
GA','SE','HIDE','HT','HIN',' ','HINI','HUN','HINV','ERT','HIN',COMA 28
HA','HAT','VEC','HMOV','E','HML','T','HOVE',' ','HRAI','SE','HSCOMA 29
I'','LAR','HSD',' ','HTRA','NS','HULT',' ','HULT','IFLY', COMA 30
J'HVEC','DIAD', COMA 31
J'HVEC','HAT','HZER','O','HEGE','XP','OROE','O','PARP','ROO','PACOMA 32
KRS','UR','PROG','UCT','PROH','ATE','ARIS','E','REAO',' ','HESE',COMA 33
L'','RHS',' ','ROH',' ','ROHS','UN','OCL','AR','SET',' ','SHOR',COMA 34
AN','SIN',' ','SIND',' ','SINH',' ','SINT',' ','SRT',' ','SUB',COMA 35
M'SUBT','RAT','SUN',' ','TAN',' ','TAND',' ','TANH',' ','TTP',COMA 36
O'TTX',' ','TYZ',' ','YORH','P','YORN','X','YORN','Z',COMA 37
NOUN=4 COMA 38
NBSIZE=141 COMA 39
NROWS=NBSIZE/7 COMA 40
NLEFT=NBSIZE-NROWS*7 COMA 41
CALL CRRAS(100) COMA 42
CALL CROPLY(' COMANDS CURRENTLY IMPLEMENTED IN OMNICONA 43
1TAD',50,41000) COMA 44
N1=NLEFT+1 COMA 45
DO 50 I=N1,7 COMA 46
50 K(I)=0 COMA 47
IF(NLEFT.EQ.0) GO TO 75 COMA 48
60 DO 70 J=1,NLEFT COMA 49
70 K(I)=1 COMA 50
75 NLOC=2 COMA 51
DO 200 J=1,NROWS COMA 52
NLOC=NLOC+1 COMA 53
DO 100 I=1,7 COMA 54
NWORK(2*I)=NAMES(NLOC)
NWORK(2*I-1)=NAMES(NLOC-1)

```

100	NLOC=NLOC+2*(NROWS+K(I))	CONA	55
	WRITE(NDUH,101) NHORK	CONA	56
101	FORMAT(7(2A4.2X))	CONA	57
	CALL FETCHTEXT(NCF,41000)	CONA	58
	CALL GROPY(TEXT,NCF,41000)	CONA	59
200	NLOC=NLOC+2	CONA	60
	IF(NLEFT.EQ.0) GO TO 500	CONA	61
	NLOC=NLOC	CONA	62
	DO 900 I=1,NLEFT	CONA	63
	NHORK(2*I)=NHORG(NLOC)	CONA	64
	NHORK(2*I-1)=NHORG(NLOC-1)	CONA	65
300	NLOC=NLOC+2*(NROWS+K(I))	CONA	66
	NL2=NLEFT+2	CONA	67
	WRITE(NDUH,101) (NHORK(I),I=1,NL2)	CONA	68
	CALL FETCHTEXT(NCF,41000)	CONA	69
	CALL GROPY(TEXT,NCF,41000)	CONA	70
500	CALL GROPY(' ',1,41000)	CONA	71
	CALL GROPY(' ',1,41000)	CONA	72
	CALL GROPY(' ',1,41000)	CONA	73
	CALL GROPY(' ',1,41000)	CONA	74
	CALL GROPY(' ',1,41000)	CONA	75
1000	RETURN 1	CONA	76
	END	CONA	77

SUBROUTINE DEFINE	DEFI 1
COMMON / BLOCK1/NOCE,M,KARO(77),KARO,ARO,ARG2,NEWCO(19),/ 77 END	DEFI 2
1,NEWCOS(19,5),KCAR,NSAVE,NFLAO	DEFI 3
COMMON / BLOCK2 / RC(2439),IARGS(69),KIND(39),AROTAD(51),NNMAX,	DEFI 4
1 NROW,NCOL,NAROS,VWXYZ(5)	DEFI 5
DIMENSION ARGS(1)	DEFI 6
EQUIVALENCE(ARGS(1),RC(2401))	DEFI 7
COMMON / BLOCK2/ NAME(4),L1,L2,J	DEFI 8
	DEFI 9
THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMAND DEFINE.	DEFI 10
	DEFI 11
IF(NAROS.GT.1.AND.NAROS.LT.5) GO TO 10	DEFI 12
CALL ERROR(10)	DEFI 13
RETURN	DEFI 14
210 CALL ERROR(20)	DEFI 15
RETURN	DEFI 16
220 CALL ERROR(11)	DEFI 17
RETURN	DEFI 18
270 CALL ERROR(9)	DEFI 19
RETURN	DEFI 20
9 CALL ERROR(3)	DEFI 21
RETURN	DEFI 22
10 K1=KIND(1)	DEFI 23
IF(K1.EQ.1.AND.NAROS.EQ.4) GO TO 210	DEFI 24
L2=NAROS+K1	DEFI 25
	DEFI 26
CHECK AND CALCULATE WORKSHEET ENTRY LOCATION INTO L	DEFI 27
	DEFI 28
CALL ADDRESS(NAROS,L)	DEFI 29
IF(L) 210,220,20	DEFI 30
20 IF(L2.NE.4.AND.NMAX.EQ.0) GO TO 270	DEFI 31
IF(L2-3) 50,40,30	DEFI 32
30 L1=NAROS-1	DEFI 33
J=IARGS(L1)	DEFI 34
IF(KIND(L1).EQ.1.OR.J.LT.1.OR.J.GT.NROW) GO TO 9	DEFI 35
L=L+J-1	DEFI 36
40 IF(K1.EQ.1) GO TO 00	DEFI 37
L1=2	DEFI 38
GO TO 55	DEFI 39
00 L1=1	DEFI 40
55 CALL ADDRESS(L1,J)	DEFI 41
IF(L1.EQ.1) GO TO 80	DEFI 42
L1=IARGS(L1)	DEFI 43
IF(K1.EQ.1.OR.L1.LT.1.OR.L1.GT.NROW) GO TO 9	DEFI 44
J=J+L1-1	DEFI 45
ARGS(L1)=RC(J)	DEFI 46
60 CALL PLAK	DEFI 47
IF(NFLAO.EQ.1) RETURN	DEFI 48
IF(L2-3) 65,70,50	DEFI 49
65 GO TO L1=1,NMAX	DEFI 50
RC(L1)=RC(J)	DEFI 51
J=J+1	DEFI 52
70 L=L+J	DEFI 53
RETURN	DEFI 54

```
00 CALL VECTOR(AROS(1),L)
   RETURN
90 RC(L)=AROS(1)
   RETURN
   END
```

```
DEFI 58
DEFI 58
DEFI 57
DEFI 58
DEFI 58
```

```

SUBROUTINE DISPLY(M)                                DISP 1
C                                                     DISP 2
C THIS SUBROUTINE IS USED TO PRODUCE THE INITIAL DISPLAY DISP 3
C ON THE 2250 WHEN EXECUTING AN OMNITAB PROGRAM.    DISP 4
C                                                     DISP 5
CALL OERAS(100)                                     DISP 6
CALL OROPLY('THIS PROGRAM IS DESIGNED TO ENABLE YOU TO USE OMNITABDISP 7
1 COMMANDS ENTERED '.72.41000)                      DISP 8
CALL OROPLY('THROUGH THE TYPEWRITER KEYBOARD DIRECTLY IN FRONT OF DISP 9
YOU. TO SIGNAL '.72.41000)                          DISP 10
CALL OROPLY('COMPLETION OF YOUR COMMAND. FIRST DEPRESS THE "ALT" KDISP 11
KEY, AND WHILE '.72.41000)                          DISP 12
CALL OROPLY('HOLDING IT DOWN, DEPRESS THE "5" KEY.'.37.41000) DISP 13
CALL OROPLY(' '.1.41000)                            DISP 14
CALL OROPLY('AT ANY TIME YOU MAY LOOK AT THE WORKSHEET BY PRESSINGDISP 15
1 ANY OF TWELVE '.72.41000)                        DISP 16
CALL OROPLY('PROGRAMMED FUNCTION KEYS. EACH KEY WILL CAUSE A 40 RDISP 17
1Y 6 SECTION OF THE '.72.41000)                    DISP 18
CALL OROPLY('WORKSHEET TO BE DISPLAYED. KEYS 4 THROUGH 9 WILL DISDISP 19
PLAY THE FIRST 40 '.72.41000)                      DISP 20
CALL OROPLY('ROWS WITH KEY 4 DISPLAYING COLUMNS 1 THROUGH 6. KEY SDISP 21
1 DISPLAYING COLUMNS'.72.41000)                  DISP 22
CALL OROPLY('0 THROUGH 10, ETC. KEYS 10 THROUGH 15 WILL LIKENISE DISP 23
10 DISPLAY THE LAST '.72.41000)                   DISP 24
CALL OROPLY('40 ROWS.'.0.41000)                    DISP 25
CALL OROPLY(' '.1.41000)                            DISP 26
CALL OROPLY('AFTER SEEING A PARTICULAR SECTION YOU MAY SEE ANOTHERDISP 27
1 SECTION BY '.72.41000)                          DISP 28
CALL OROPLY('PRESSING ANOTHER KEY OR YOU MAY ENTER MORE OMNITAB COORDISP 29
INATES THROUGH THE '.72.41000)                    DISP 30
CALL OROPLY('TYPEWRITER KEYBOARD.'.20.41000)       DISP 31
CALL OROPLY(' '.1.41000)                            DISP 32
CALL OROPLY('BY PRESSING KEY 30 YOU WILL RETURN TO THIS DISPLAY.  OP 33
BY PRESSING KEY 31 '.72.41000)                    DISP 34
CALL OROPLY('YOU WILL TERMINATE THIS PROGRAM. BY PRESSING KEY 3 YDISP 35
OU WILL BE ABLE TO '.72.41000)                    DISP 36
CALL OROPLY('SEE A DISPLAY OF THE OMNITAB COMMANDS CURRENTLY AVAILDISP 37
ABLE. BY PRESSING '.72.41000)                     DISP 38
CALL OROPLY('KEY 2 YOU WILL SEE A LIST OF THE OMNITAB COMMANDS PAIDISP 39
1CH YOU HAVE ENTERED'.72.41000)                   DISP 40
CALL OROPLY(' '.1.41000)                            DISP 41
CALL OROPLY(' '.1.41000)                            DISP 42
CALL OROPLY(' '.1.41000)                            DISP 43
CALL OROPLY(' '.1.41000)                            DISP 44
CALL OROPLY(' '.1.41000)                            DISP 45
CALL OROPLY(' '.1.41000)                            DISP 46
1000 RETURN 1                                       DISP 47
END

```

SUBROUTINE EOBINT	EOBI	1
COMMON KEY,IOVLY,ITYPE	EOBI	2
ITYPE=2	EOBI	3
CALL CPOST	EOBI	4
RETURN	EOBI	5
END	EOBI	6
SUBROUTINE ERASE	ERAS	1
COMMON / BLOCKR/RODE,H,KARD(77),KARG,ARG,ARG2,NEHCD(19),KROEND	ERAS	2
1,NEHCD(19,5),KSAVE,NSAVE,NFLAO	ERAS	3
COMMON / BLOCKO / RC(2499),IAROS(69),KIND(39),AROTAB(51),NRMAX,	ERAS	4
1 NRON,NCOL,NAROS,VHXY2(5)	ERAS	5
	ERAS	6
C THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMAND ERASE.	ERAS	7
C	ERAS	8
C	ERAS	9
CALL CHKCOL(1)	ERAS	10
IF(1.EQ.0.OR.NAROS.EQ.0) GO TO 30	ERAS	11
CALL ERROR(3)	ERAS	12
20 RETURN	ERAS	13
30 CALL PLOK	ERAS	14
IF(NFLAO.EQ.1) RETURN	ERAS	15
IF(NAROS .EQ. 0) GO TO 50	ERAS	16
IF(NRMAX.EQ.0) GO TO 20	ERAS	17
DO 40 I = 1, NAROS	ERAS	18
40 CALL VECTOR(0., IAROS(I))	ERAS	19
GO TO 20	ERAS	20
C	ERAS	21
C	ERAS	22
C CLEAR ALL OF DIMENSIONED WORKSHEET.	ERAS	23
50 NRMAX = NRON = NCOL	ERAS	24
CALL VECTOR(0., 1)	ERAS	25
NRMAX = 0	ERAS	26
GO TO 20	ERAS	27
END		

	SUBROUTINE ERROR(1)	ERRO	1
	COMMON / BLOCKA/MODE,M,KARD(77),KARG,ARG,ARG2,NEWCD(19),KROEND	ERRO	2
	1,NEWCDG(10,5),KSAVE,NSAVE,NFLAG	ERRO	3
	COMMON/KPLOT/NFRAME,KKND,SIZE,SPACE	ERRO	4
	COMMON KEY,IOVLY,ITYPE	ERRO	5
C		ERRO	6
C	THIS SUBROUTINE IS USED TO DESPLAY ERROR MESSAGES.	ERRO	7
C	I IS USED TO INDICATE WHICH MESSAGE IS TO BE DISPLAYED.	ERRO	8
C		ERRO	9
	IF(NFLAG.EQ.1) RETURN	ERRO	10
	J=(1-100)/50	ERRO	11
	IF(J.EQ.0) GO TO 200	ERRO	12
	CALL GROPY(' ',1,41010)	ERRO	13
1020	CALL GORSP(8)	ERRO	14
	CALL GROPY(NEWCD,76,41000)	ERRO	15
	CALL GROPY(' ',1,41000)	ERRO	16
	IF(J.GT.0) GO TO 400	ERRO	17
	NFLAG = 1	ERRO	18
	GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,	ERRO	19
	1 23,24,25,26,27,28,29,30), I	ERRO	20
	1 CALL GROPY('NAME NOT FOUND IN LIBRARY',26,41000)	ERRO	21
	2 CONTINUE	ERRO	22
	GO TO 5000	ERRO	23
	3 CALL GROPY('ILLEGAL ARGUMENT ON CARD',24,41000)	ERRO	24
	4 CONTINUE	ERRO	25
	5 CONTINUE	ERRO	26
	6 CONTINUE	ERRO	27
	GO TO 5000	ERRO	28
	7 CALL GROPY('ILLEGAL STATEMENT',17,41000)	ERRO	29
	GO TO 5000	ERRO	30
	8 CALL GROPY('CONSTANT NOT IN TABLE',21,41000)	ERRO	31
	GO TO 5000	ERRO	32
	9 CALL GROPY('NMAX = 0',9,41000)	ERRO	33
	GO TO 5000	ERRO	34
	10 CALL GROPY('ILLEGAL NUMBER OF ARGUMENTS',27,41000)	ERRO	35
	GO TO 5000	ERRO	36
	11 CALL GROPY('COLUMN NUMBER TOO BIG OR LESS THAN 1 ',37,41000)	ERRO	37
	12 CONTINUE	ERRO	38
	13 CONTINUE	ERRO	39
	14 CONTINUE	ERRO	40
	15 CONTINUE	ERRO	41
	GO TO 5000	ERRO	42
	16 CALL GROPY('ILLEGAL SIZE ROW NUMBER ',24,41000)	ERRO	43
	GO TO 5000	ERRO	44
	17 CALL GROPY('DEFINED MATRIX OVERFLOWS WORKSHEET',34,41000)	ERRO	45
	GO TO 5000	ERRO	46
	18 CALL GROPY('INTEGER ARGUMENT LESS THAN -0101',32,41000)	ERRO	47
	19 CONTINUE	ERRO	48
	GO TO 5000	ERRO	49
	20 CALL GROPY('IMPROPER TYPE OF ARGUMENT',25,41000)	ERRO	50
	GO TO 5000	ERRO	51
	21 CALL GROPY('ASTERISK SIGNIFYING IMPLYING THROUGH INCORRECT',42,41000)	ERRO	52
	22 CONTINUE	ERRO	53
	GO TO 5000	ERRO	54

23	CALL CROPLY('MATRIX IS TOO LARGE TO INVERT USING THIS INTERACTIVE	ERRO	55
	10DEVICE',59.41000)	ERRO	56
3000	CALL CROPLY('PLEASE SUBMIT AS A BATCH JOB',28.41000)	ERRO	57
	GO TO 1000	ERRO	58
24	CALL CROPLY('PRODUCT OF MATRIX MULTIPLICATION IS TOO LARGE FOR INTERRO	59	
	1ERACTIVE MODE',66.41000)	ERRO	60
	GO TO 3000	ERRO	61
25	CONTINUE	ERRO	62
26	CONTINUE	ERRO	63
27	CONTINUE	ERRO	64
28	CONTINUE	ERRO	65
29	CONTINUE	ERRO	66
30	CONTINUE	ERRO	67
5000	CALL CROPLY('PLEASE REENTER ',15.41000)	ERRO	68
1000	RETURN	ERRO	69
1010	CALL GENAS(100)	ERRO	70
	GO TO 1020	ERRO	71
C		ERRO	72
C	ARITHMETIC TROUBLES	ERRO	73
C		ERRO	74
200	NFLAG=1	ERRO	75
	CALL AERR(1-100)	ERRO	76
	KSAVE=KSAVE+1	ERRO	77
	DO 9001 JJJ=1,19	ERRO	78
9001	NEWCOS(JJJ,KSAVE)=NEWCO(JJJ)	ERRO	79
	IF(KSAVE.LT.5) GO TO 250	ERRO	80
	IF(10VLY.GT.40) GO TO 9003	ERRO	81
9004	WRITE(KSAVE,10VLY) NEWCOS	ERRO	82
	KSAVE=0	ERRO	83
250	RETURN	ERRO	84
9003	CALL PRORAN(1)	ERRO	85
	IF(KEY.EQ.91) RETURN	ERRO	86
	10VLY = 1	ERRO	87
	GO TO 9004	ERRO	88
C		ERRO	89
C	INFORMATIVE DIAGNOSTIC	ERRO	90
C		ERRO	91
400	II=1-200	ERRO	92
	GO TO (401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412,	ERRO	93
	1 413, 414, 415), II	ERRO	94
401	CALL CROPLY('TOO MUCH DATA',13.41000)	ERRO	95
402	CONTINUE	ERRO	96
403	CONTINUE	ERRO	97
404	CONTINUE	ERRO	98
405	CONTINUE	ERRO	99
406	CONTINUE	ERRO	100
407	CONTINUE	ERRO	101
408	CONTINUE	ERRO	102
409	CONTINUE	ERRO	103
410	CONTINUE	ERRO	104
	GO TO 4000	ERRO	105
411	CONTINUE	ERRO	106
412	CONTINUE	ERRO	107
413	CONTINUE	ERRO	108

414	CONTINUE	ERRO 109
415	CONTINUE	ERRO 110
4000	CALL ORDPY('PRESS KEY 1 FOR STANDARD FIXUP OR KEY 2 TO CANCEL EXEENRO 111	
	ICUTION.' .60.41000)	ERRO 112
4500	CALL GWAIT	ERRO 119
	IF(IITYPE.NE.1) GO TO 4500	ERRO 114
	IF(KEY.EQ.1) GO TO 1000	ERRO 118
	IF(KEY.EQ.2.(R.KEY.EQ.31) GO TO 4550	ERRO 116
	IF(KEY.NE.22) GO TO 4500	ERRO 117
	CALL SCOPLT (0,NFRAME,KKNO.SIZE.SPACE.INTEQ.IRCODE)	ERRO 118
	CALL SCOPLT (1,IQUM)	ERRO 119
	CALL OCALM	ERRO 120
	GO TO 4500	ERRO 121
4550	NFLAG=1	ERRO 122
	GO TO 1000	ERRO 123
	END	ERRO 124

	SUBROUTINE EXCHNG	EXCH 1
	COMMON / BLOCKA/NODE,N,KARD(77),KARG,ARG,ARC2,NEWCD(19),KROEND	EXCH 2
	1,NEWCD5(19,5),KSAVE,NSAVE,NFLAG	EXCH 3
	COMMON / BLOCKD / RC(2439),IARGS(69),KIND(39),ARCTAB(51),NRMAX,	EXCH 4
	1 NROW,NCOL,NARCS,VNXYZ(5)	EXCH 5
		EXCH 6
C	THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMAND EXCHANGE.	EXCH 7
C		EXCH 8
C	IF(NARCS.NE.(NARCS/2)*2.OR.NARCS.EQ.0) GO TO 910	EXCH 9
	CALL CHKCOL(1)	EXCH 10
	IF(1.EQ.1) GO TO 909	EXCH 11
	IF(NRMAX.LT.1) GO TO 909	EXCH 12
	CALL PLBK	EXCH 13
	IF(NFLAG.EQ.1) RETURN	EXCH 14
	DO 90 I=1,NARCS,2	EXCH 15
	DO 90 N=1,NRMAX	EXCH 16
	JJ=IARGS(I)+N-1	EXCH 17
	KK=IARGS(I+1)+N-1	EXCH 18
	WORK=RC(JJ)	EXCH 19
	RC(JJ)=RC(KK)	EXCH 20
	RC(KK)=WORK	EXCH 21
	90 CONTINUE	EXCH 22
	GO TO 999	EXCH 23
	909 CALL ERROR (9)	EXCH 24
	GO TO 999	EXCH 25
	910 CALL ERROR (10)	EXCH 26
	GO TO 999	EXCH 27
	999 CALL ERROR (9)	EXCH 28
	999 RETURN	EXCH 29
	END	EXCH 30

```

SUBROUTINE EXPAND( J, WHERE )
COMMON / BLOCK / RC(2499), IARG(99), KIND(99), AROT(51), NRMAY,
1 NROW, NCOL, NARG, VXYZ(5)
COMMON / BLOCK / NAME(4), L1, L2, ISRFLO
DIMENSION ARG(39)
EQUIVALENCE( ARG(1), RC(2401) )
DIMENSION WHERE( 1 )
C
C THIS SUBROUTINE TAKES THE INFORMATION STORED IN THE FIRST J
C POSITIONS OF THE ARRAY WHERE AND CONVERTS THIS INFORMATION INTO A
C FORM WHICH CAN BE EASILY USED BY THE ORNITAB COMMANDS.
I = 0
J = J
JJJ = J
10 I = I + 1
15 I = I + 1
IF( I .GE. JJJ ) GO TO 45
T = WHERE( I )
IF( T ) 40, 30, 20
20 KIND( II ) = 0
IARG( II ) = T - 0102.
GO TO 10
30 KIND( II ) = 1
I = I + 1
ARG( II ) = WHERE( I )
GO TO 10
40 IF( T .EQ. -1. ) GO TO 100
CALL XPND( WHERE( I ), K, AROT( II ), KND )
IF( K .GE. 0 ) GO TO 50
41 K = - K
42 CALL ERROR( K )
43 RETURN
50 KIND( II ) = KND
IF( KND .EQ. 0 ) IARG( II ) = AROT( II )
I = I + 1
GO TO 10
C
C EXPAND FROM PREVIOUS INTEGER ARGUMENT TO FOLLOWING.
C
100 I = I + 1
C PICK UP NEXT ARG
IU = WHERE( I )
IF( KIND( II-1 ) .NE. 0 .OR. I .GE. J ) GO TO 200
IF( IU ) 100, 200, 105
105 IU = IU - 0102
100 K = IU - IARG( II-1 )
NARG = NARG + IARG( K ) - 1
IF( K ) 110, 10, 120
110 INC = -1
K = -K
GO TO 140
120 INC = 1
140 DO 100 IT = 1, K
KIND( II ) = 0
EXPAN 1
EXPAN 2
EXPAN 3
EXPAN 4
EXPAN 5
EXPAN 6
EXPAN 7
EXPAN 8
EXPAN 9
EXPAN 10
EXPAN 11
EXPAN 12
EXPAN 13
EXPAN 14
EXPAN 15
EXPAN 16
EXPAN 17
EXPAN 18
EXPAN 19
EXPAN 20
EXPAN 21
EXPAN 22
EXPAN 23
EXPAN 24
EXPAN 25
EXPAN 26
EXPAN 27
EXPAN 28
EXPAN 29
EXPAN 30
EXPAN 31
EXPAN 32
EXPAN 33
EXPAN 34
EXPAN 35
EXPAN 36
EXPAN 37
EXPAN 38
EXPAN 39
EXPAN 40
EXPAN 41
EXPAN 42
EXPAN 43
EXPAN 44
EXPAN 45
EXPAN 46
EXPAN 47
EXPAN 48
EXPAN 49
EXPAN 50
EXPAN 51
EXPAN 52
EXPAN 53
EXPAN 54

```

```

      IARGC( II ) = IARGC( II-1 ) + INC
150  II = II + 1
      GO TO 15
160  CALL XPND( WHERE( I ) , K , IARGC( II ) , KND )
      IF( K .LT. 0 ) GO TO 41
      I = I + K
      IF( KND .EQ. 0 ) GO TO 170
      K = 20
      GO TO 42
170  IU = IARGC( II )
      GO TO 106
200  K=21
      GO TO 42
      END

```

```

EXPA 55
EXPA 56
EXPA 57
EXPA 58
EXPA 59
EXPA 60
EXPA 61
EXPA 62
EXPA 63
EXPA 64
EXPA 65
EXPA 66
EXPA 67
EXPA 68

```

SUBROUTINE EXPCON	EXPC	1
COMMON / BLOCKA/MODE,M,KARD(77),KARG,ARG,ARG2,NEWCD(19),KROEND	EXPC	2
1,NEWCOS(10,5),KSAVE,NSAVE,NFLAO	EXPC	3
COMMON / BLOCKD / RC(2+99),IARG(69),KIND(39),ARGTAB(51),NRMAX,	EXPC	4
1,NROW,NCOL,NARGO,VWXYZ(5)	EXPC	5
COMMON/BLOCKE/NRHE(4),L1,L2,ISRFLO	EXPC	6
COMMON/SCRAT/A(80)	EXPC	7
***** THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS	EXPC	8
***** HVECDIRO, HVECDIRO, HVECHAT, AVECARR, MHATVEC AND AARVEC.	EXPC	9
IF(IARG(1).LT.5.OR.IARG(2).GT.6) GO TO 10	EXPC	10
IF(L2.GE.5) GO TO 300	EXPC	11
100 J=IARG(1)	EXPC	12
CALL CKIND(J)	EXPC	13
IF(J.NE.0) GO TO 3	EXPC	14
J=1	EXPC	15
CALL HTXCHX(J)	EXPC	16
IF(J-1) 102,3,17	EXPC	17
102 CALL ADRESS(IARG(1),ILL)	EXPC	18
IF(ILL.LE.0) GO TO 11	EXPC	19
IH=IARG(1)	EXPC	20
ILC=ILL	EXPC	21
IL=IARG(3)	EXPC	22
KHX=IARG(4)	EXPC	23
NHX=IARG(5)	EXPC	24
IF(L2.GT.2) IL=MIND(IL,IARG(4),00)	EXPC	25
IF(IARG(6).NE.6) GO TO 103	EXPC	26
ILC=ILC+IARG(6)-1	EXPC	27
IL=MIND(IL,61-IARG(5))	EXPC	28
103 IXX=ILC+IL-1	EXPC	29
IF(IXX.GT.ILL+NROW-1) GO TO 3	EXPC	30
CALL PLOK	EXPC	31
IF(INFLAG.EQ.1) RETURN	EXPC	32
IF(L2.GE.5) GO TO 310	EXPC	33
IF(L2.GE.3) GO TO 220	EXPC	34
***** VEC DIAO	EXPC	35
120 DO 125 IC=1,IL	EXPC	36
A(IC)=RC(IH)	EXPC	37
125 IH=IH+1-NROW	EXPC	38
IH=1	EXPC	39
DO 130 I=ILC,IXX	EXPC	40
RC(I)=A(IH)	EXPC	41
130 IH=IH+1	EXPC	42
RETURN	EXPC	43
***** VECTHAT	EXPC	44
220 IC=1	EXPC	45
IHH=IH	EXPC	46
DO 240 J=1,NHX	EXPC	47
IK=IHH	EXPC	48
DO 230 Im=1,NHX	EXPC	49
IF(IC.GT.IL) GO TO 245	EXPC	50
A(IC)=RC(IH)	EXPC	51
IH=IH+NROW	EXPC	52
230 IC=IC+1	EXPC	53
240 IHH=IHH+1	EXPC	54

```

245 DO 250 I=1,IL
    RC(ILC)=A(I)
250 ILC=ILC+1
    RETURN
C===== MATVEC
300 IL=IARG6(1)
    IC=2
    IF (NARGS.NE.6) GO TO 302
    IC=3
    ILL=IARG6(2)
302 DO 305 I=1,4
    IARG6(I)=IARG6(IC)
305 IC=IC+1
    IARG6(5)=IL
    IARG6(6)=ILL
    GO TO 100
310 DO 320 I=1,IL
    A(I)=RC(ILC)
320 ILC=ILC+1
    ILC = 1
    DO 340 J=1,NMX
    IMC=JM
    DO 330 J=1,KMX
    RC(JMC)=A(ILC)
    JMC=JMC+NROW
330 ILC=ILC+1
340 JM=JM+1
    RETURN
3 CALL ERROR(3)
    RETURN
10 CALL ERROR(10)
    RETURN
17 CALL ERROR(17)
    RETURN
11 CALL ERROR(11)
    RETURN
END

```

```

EXPC 55
EXPC 56
EXPC 57
EXPC 58
EXPC 59
EXPC 60
EXPC 61
EXPC 62
EXPC 63
EXPC 64
EXPC 65
EXPC 66
EXPC 67
EXPC 68
EXPC 69
EXPC 70
EXPC 71
EXPC 72
EXPC 73
EXPC 74
EXPC 75
EXPC 76
EXPC 77
EXPC 78
EXPC 79
EXPC 80
EXPC 81
EXPC 82
EXPC 83
EXPC 84
EXPC 85
EXPC 86
EXPC 87
EXPC 88
EXPC 89
EXPC 90
EXPC 91

```


	FUNCTION FCOS(X)	FCOS	1
C	THIS FUNCTION SUBPROGRAM CALCULATES THE COSINE OF THE ARGUMENT X.	FCOS	2
	IF(ABS(X) .GT. .DEG) GO TO 2	FCOS	3
	FCOS = COS(X)	FCOS	4
	1 RETURN	FCOS	5
	2 CALL ERROR(104)	FCOS	6
	FCOS = 0.	FCOS	7
	GO TO 1	FCOS	8
	END	FCOS	9
	FUNCTION FEXP(X)	FEXP	1
C	THIS FUNCTION SUBROUTINE CALCULATES EXP(X).	FEXP	2
	IF(ABS(X).GT.174.) GO TO 2	FEXP	3
	FEXP = EXP(X)	FEXP	4
	1 RETURN	FEXP	5
	2 CALL ERROR(102)	FEXP	6
	FEXP = 0.	FEXP	7
	GO TO 1	FEXP	8
	END	FEXP	9
	FUNCTION FEXP2(B, E)	FEXP	1
C	THIS FUNCTION SUBPROGRAM CALCULATES B**E IF SUCH AN OPERATION	FEXP	2
C	DOESN'T PRODUCE OVERFLOW OR UNDERFLOW.	FEXP	3
C	IE = E	FEXP	4
	IF(E .EQ. FLOAT(IE)) GO TO 2	FEXP	5
	IF(0.GT.0.) GO TO 3	FEXP	6
	CALL ERROR(101)	FEXP	7
	FEXP2=0.	FEXP	8
	RETURN	FEXP	9
	3 FEXP2 = FEXP(E = AL00(B))	FEXP	10
	1 RETURN	FEXP	11
	2 FEXP2 = 0 .AND. IE	FEXP	12
	GO TO 1	FEXP	13
	END	FEXP	14
		FEXP	15
		FEXP	16

SUBROUTINE FLIP	FLIP 1
COMMON / BLOCKA/MODE,N,KARD(77),KARD,ARC,ARC2,NENCO(19),KROEND	FLIP 2
1,NENCDS(19,5),KSAVE,NSAVE,NFLAD	FLIP 3
COMMON / BLOCKD / RC(2433),IARCS(99),MINU(99),ARCTAB(51),NRMAX,	FLIP 4
1 NRUN,NCDL,NARCS,VWXYZ(5)	FLIP 5
C THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMAND FLIP.	FLIP 6
C	FLIP 7
C	FLIP 8
IF(NARCS .GT. 0 .AND. MOD(NARCS, 2) .EQ. 0) GO TO 20	FLIP 9
I = 10	FLIP 10
10 CALL ERROR(1)	FLIP 11
15 RETURN	FLIP 12
20 CALL CHKCOL(1)	FLIP 13
IF(1 .EQ. 0) GO TO 25	FLIP 14
I=3	FLIP 15
GO TO 10	FLIP 16
25 IF(NRMAX-GE.1) GO TO 30	FLIP 17
I=9	FLIP 18
GO TO 10	FLIP 19
30 CALL PLOK	FLIP 20
IF(NFLAD.EQ.1.OR.NRMAX.EQ.1) RETURN	FLIP 21
KK = NRMAX - 1	FLIP 22
K = KK / 2	FLIP 23
DO 00 I = 1, NARCS, 2	FLIP 24
N = IARCS(I)	FLIP 25
N = IARCS(I+1)	FLIP 26
NN = N + KK	FLIP 27
NN = N + KK	FLIP 28
NNN = N + K	FLIP 29
DO 50 J = N, NNN	FLIP 30
A = RC(J)	FLIP 31
RC(N) = RC(NN)	FLIP 32
RC(NN) = A	FLIP 33
N = N + 1	FLIP 34
NN = NN - 1	FLIP 35
NN = NN - 1	FLIP 36
60 CONTINUE	FLIP 37
GO TO 15	FLIP 38
END	FLIP 39

C	FUNCTION FLOG(X)	FLOG	1
C		FLOG	2
C	THIS FUNCTION SUBPROGRAM CHECKS X AND IF IT IS POSITIVE	FLOG	3
C	CALCULATES THE NATURAL LOG OF X.	FLOG	4
		FLOG	5
	IF(X .GT. 0.) GO TO 1	FLOG	6
	CALL ERROR(101)	FLOG	7
	FLOG = 0.	FLOG	8
	GO TO 2	FLOG	9
1	FLOG = ALOG(X)	FLOG	10
2	RETURN	FLOG	11
	END	FLOG	12
	FUNCTION FSIN(X)	FSIN	1
C		FSIN	2
C	THIS FUNCTION SUBPROGRAM CALCULATES THE SINE OF X.	FSIN	3
C		FSIN	4
	IF(ABS(X) .GT. .8EO) GO TO 2	FSIN	5
	FSIN = SIN(X)	FSIN	6
1	RETURN	FSIN	7
2	CALL ERROR(104)	FSIN	8
	FSIN = 0.	FSIN	9
	GO TO 1	FSIN	10
	END	FSIN	11
	FUNCTION FSQRT(X)	FSQR	1
C		FSQR	2
C	THIS FUNCTION SUBPROGRAM CHECKS X AND IF IT IS POSITIVE	FSQR	3
C	CALCULATES THE SQUARE ROOT OF X.	FSQR	4
		FSQR	5
	IF(X .LT. 0.) GO TO 2	FSQR	6
	FSQR = SQRT(X)	FSQR	7
1	RETURN	FSQR	8
2	CALL ERROR(101)	FSQR	9
	FSQR = 0.	FSQR	10
	GO TO 1	FSQR	11
	END	FSQR	12

```

SUBROUTINE FUNCT
COMMON / BLOCKA/MODE,M,KARD(77),KARO,ARC,AR02,NENCO(18),KRDEND
1.NEVCOS(19.5),KSAVE,NCAVE,NFLAQ
COMMON / BLOCKD / RC(2499),IARCS(99),KIND(99),ARCTAB(51),NRMAX,
2 NRON,NCOL,NARCS,VWXYZ(5)
COMMON/BLOCKE/NAME(4),L1,L2,ISRFLO
COMMON/CONSTS/PI,E,HALFPI,DEG,RAD
REAL*8 XO
DIMENSION II( 2 )
EQUIVALENCE ( II, II( 1 ) ), ( I2, II( 2 ) )
DIMENSION ARCS(1)
EQUIVALENCE(ARCS(1),RC(2401))
C
C THIS SUBROUTINE IS CALLED FOR THE FOLLOWING COMMANDS: SIN, COS,
C TAN, COT, ARCSIN, ASIN, ARCCOS, ACOS, ARCTAN, ATAN, ARCCOT, ACOT,
C SIND, COSD, TAND, COTD, ASIND, ACOSD, ATAND, ACOTD, ABS,
C ABSOLUTE, EXP, EXPONENT, LOG, LOGE, SORT, NEGEXP, LOGTEN,
C ANTILOG, SINH, COSH, TANH, COTH, ASINH, ACOSH, ATANH, ACOTH AND
C DEYNOR.
C
IF( NARCS .EQ. 2 .OR. NARCS .EQ. 3 ) GO TO 10
CALL ERROR( 10 )
GO TO 200
10 CALL ADDRESS( NARCS, IL )
IF( IL ) 20, 30, 40
20 CALL ERROR( 20 )
GO TO 200
30 CALL ERROR( 11 )
GO TO 200
40 IL2 = IL + NRMAX - 1
NARCS = NARCS - 1
DO 60 I = 1, NARCS
CALL ADDRESS( I, II( 1 ) )
IF( II( 1 ) ) 45, 30, 50
45 II( 1 ) = -II( 1 )
50 CONTINUE
IF(NAME(1).NE.3050.OR.NAME(4).NE.13262) GO TO 51
IF(L2.GT.19.AND..2.LT.28) L2=L2+8
IF(L2.GT.35.AND..12.LT.40) L2=L2-4
51 IF( KIND( 1 ) .EQ. 0 ) GO TO 55
X = ARCS( 1 )
LOCRTN = 1
DO 101 460,480,500,470,490,510,520,530,540,550,560,570,580,590,600,
1 610,460,480,500,300,310,320,330,340,350,360,370,380,390,400,
2 410,420,430,440,450,340,350,360,370,1,2
52 ARCS( 1 ) = X
55 IF( NRMAX .NE. 0 ) GO TO 60
CALL ERROR( 0 )
GO TO 200
60 IF(INFLAG.EQ.0) CALL PL6K
IF(INFLAG.EQ.1) GO TO 200
ARCS(1) TO INDEX
IF( NARCS .EQ. 2 ) GO TO 90
IF( KIND( 1 ) .EQ. 0 ) GO TO 70

```

```

FUNC 1
FUNC 2
FUNC 3
FUNC 4
FUNC 5
FUNC 6
FUNC 7
FUNC 8
FUNC 9
FUNC 10
FUNC 11
FUNC 12
FUNC 13
FUNC 14
FUNC 15
FUNC 16
FUNC 17
FUNC 18
FUNC 19
FUNC 20
FUNC 21
FUNC 22
FUNC 23
FUNC 24
FUNC 25
FUNC 26
FUNC 27
FUNC 28
FUNC 29
FUNC 30
FUNC 31
FUNC 32
FUNC 33
FUNC 34
FUNC 35
FUNC 36
FUNC 37
FUNC 38
FUNC 39
FUNC 40
FUNC 41
FUNC 42
FUNC 43
FUNC 44
FUNC 45
FUNC 46
FUNC 47
FUNC 48
FUNC 49
FUNC 50
FUNC 51
FUNC 52
FUNC 53
FUNC 54

```

C		FUNC 55
C	TWO ARGUMENTS. FIRST IS A CONSTANT	FUNC 56
C	CALL VECTOR(ARG(1), IL)	FUNC 57
	GO TO 200	FUNC 58
C		FUNC 59
C	TWO ARGUMENTS. FIRST IS A COLUMN NUMBER	FUNC 60
C		FUNC 61
	70 LOCRTN = 2	FUNC 62
	I = IL	FUNC 63
	80 IF (I .GT. ILZ) GO TO 200	FUNC 64
	X = RC(I)	FUNC 65
	GO TO INDEX, (250,300,310,320,330,340,350,360,370,380,390,400,410,	FUNC 66
	1420,430,440,450,460,470,480,490,500,510,520,530,540,550,560,570,	FUNC 67
	2580,590,600,610)	FUNC 68
	75 RC(I) = X	FUNC 69
	I1 = I1 + 1	FUNC 70
	I=I+1	FUNC 71
	GO TO 80	FUNC 72
	90 K2 = 1 - KIND(2)	FUNC 73
	IF(KIND(1) .EQ. 0) GO TO 110	FUNC 74
C		FUNC 75
C	THREE ARGUMENTS. FIRST ONE A CONSTANT	FUNC 76
C		FUNC 77
	GO 100 I = IL, ILZ	FUNC 78
	RC(I) = RC(I) + RC(I2) * ARG(1)	FUNC 79
	100 I2 = I2 + K2	FUNC 80
	GO TO 200	FUNC 81
C		FUNC 82
C	THREE ARGUMENTS. FIRST A COLUMN NUMBER	FUNC 83
C		FUNC 84
	110 LOCRTN = 3	FUNC 85
	I=IL	FUNC 86
	120 IF (I .GT. ILZ) GO TO 200	FUNC 87
	X = RC(I)	FUNC 88
	GO TO INDEX, (250,300,310,320,330,340,350,360,370,380,390,400,410,	FUNC 89
	1420,430,440,450,460,470,480,490,500,510,520,530,540,550,560,570,	FUNC 90
	2580,590,600,610)	FUNC 91
	115 RC(I) = RC(I) + RC(I2) * X	FUNC 92
	I1 = I1 + 1	FUNC 93
	I2 = I2 + K2	FUNC 94
	I=I+1	FUNC 95
	GO TO 120	FUNC 96
	200 RETURN	FUNC 97
	250 GO TO(459,479,499,489,409,509,619,629,539,549,559,569,579,689,599,	FUNC 98
	1 609,459,479,499,299,309,319,329,339,349,359,369,379,389,399,	FUNC 99
	2 409,419,429,439,449,339,349,359,369),L2	FUNC 100
	260 CALL ERROR(L)	FUNC 101
	285 X = 0.	FUNC 102
	275 GO TO (52, 75, 115), LOCRTN	FUNC 103
C	SIN	FUNC 104
	290 ASSIGN 300 TO INDEX	FUNC 105
	300 X = FSIN(X)	FUNC 106
	GO TO 275	FUNC 107
		FUNC 108

C	COS	FUNC 109
309	ASSIGN 310 TO INDEX	FUNC 110
310	X = FCOS(X)	FUNC 111
	GO TO 275	FUNC 112
C	TAN	FUNC 113
319	ASSIGN 320 TO INDEX	FUNC 114
320	X = FSIN(X) / FCOS(X)	FUNC 115
	GO TO 275	FUNC 116
C	COT	FUNC 117
329	ASSIGN 330 TO INDEX	FUNC 118
330	IF(X.EQ.0.) X=1.E-75	FUNC 119
	X = FCOS(X) / FSIN(X)	FUNC 120
	GO TO 275	FUNC 121
C	ASIN	FUNC 122
339	ASSIGN 340 TO INDEX	FUNC 123
340	IF(ABS(X) - 1.) 341, 342, 343	FUNC 124
341	IF(ABS(X).LT.1.E-39) X=0.	FUNC 125
	X = ATAN(X / SQRT(1. - X ** 2))	FUNC 126
	GO TO 275	FUNC 127
342	X = SIGN(HALFPI, X)	FUNC 128
	GO TO 275	FUNC 129
343	L = 103	FUNC 130
	GO TO 280	FUNC 131
C	ACOS	FUNC 132
349	ASSIGN 350 TO INDEX	FUNC 133
350	IF(ABS(X).GT.1.100 TO 349	FUNC 134
	IF(ABS(X).LT.1.E-39) GO TO 342	FUNC 135
	X = ATAN(SQRT(1. - X ** 2) / X)	FUNC 136
	GO TO 275	FUNC 137
C	ATAN	FUNC 138
359	ASSIGN 360 TO INDEX	FUNC 139
360	X = ATAN(X)	FUNC 140
	GO TO 275	FUNC 141
C	ACOT	FUNC 142
369	ASSIGN 370 TO INDEX	FUNC 143
370	IF(X.EQ.0.) X=1.E-75	FUNC 144
	X = ATAN(1. / X)	FUNC 145
	GO TO 275	FUNC 146
C	SINH	FUNC 147
379	ASSIGN 380 TO INDEX	FUNC 148
380	X = DEO * X	FUNC 149
	GO TO 300	FUNC 150
C	COSH	FUNC 151
389	ASSIGN 390 TO INDEX	FUNC 152
390	X = DEO * X	FUNC 153
	GO TO 310	FUNC 154
C	TANH	FUNC 155
399	ASSIGN 400 TO INDEX	FUNC 156
400	X = DEO * X	FUNC 157
	GO TO 320	FUNC 158
C	COTO	FUNC 159
409	ASSIGN 410 TO INDEX	FUNC 160
410	IF(ABS(X).LT.1.E-74) X=1.E-74	FUNC 161
	X = DEO * X	FUNC 162

GO TO 330	FUNC 169
C ASIND	FUNC 164
419 ASSIGN 420 TO INDEX	FUNC 165
420 IF(ABS(X) - 1.) 421, 422, 343	FUNC 166
421 IF(ABS(X).LT.1.E-39) X=0.	FUNC 167
X = RAD = ATAN(X / SQRT(1. - X ** 2))	FUNC 168
GO TO 275	FUNC 169
422 X = SIGN(90., X)	FUNC 170
GO TO 275	FUNC 171
C ACOSD	FUNC 172
429 ASSIGN 430 TO INDEX	FUNC 173
430 IF(ABS(X).GT.1) GO TO 343	FUNC 174
IF(ABS(X).LT.1.E-39) GO TO 422	FUNC 175
X = RAD = ATAN(SQRT(1. - X ** 2) / X)	FUNC 176
GO TO 275	FUNC 177
C ATAND	FUNC 178
439 ASSIGN 440 TO INDEX	FUNC 179
440 X = RAD = ATAN(X)	FUNC 180
GO TO 275	FUNC 181
C ACOTD	FUNC 182
449 ASSIGN 450 TO INDEX	FUNC 183
450 IF(X.EQ.0.) X=1.E-70	FUNC 184
X = RAD = ATAN(1. / X)	FUNC 185
GO TO 275	FUNC 186
C ADS	FUNC 187
459 ASSIGN 460 TO INDEX	FUNC 188
460 X = ADS(X)	FUNC 189
GO TO 275	FUNC 190
C SORT	FUNC 191
469 ASSIGN 470 TO INDEX	FUNC 192
470 X = FSQRT(X)	FUNC 193
GO TO 275	FUNC 194
C EXP	FUNC 195
479 ASSIGN 480 TO INDEX	FUNC 196
480 X = FEXP(X)	FUNC 197
GO TO 275	FUNC 198
C HEXP	FUNC 199
489 ASSIGN 490 TO INDEX	FUNC 200
490 X = FEXP(-X)	FUNC 201
GO TO 275	FUNC 202
C LOG	FUNC 203
499 ASSIGN 500 TO INDEX	FUNC 204
500 X = FLOG(X)	FUNC 205
GO TO 275	FUNC 206
C LOG10	FUNC 207
509 ASSIGN 510 TO INDEX	FUNC 208
510 IF(X .GT. 0.) GO TO 511	FUNC 209
L = 101	FUNC 210
GO TO 200	FUNC 211
511 X = ALOG10(X)	FUNC 212
GO TO 275	FUNC 213
C ALOG	FUNC 214
519 ASSIGN 520 TO INDEX	FUNC 215
520 IF(X .GT. 75.) GO TO 522	FUNC 216

	X = 10. * X		
	GO TO 278		FUNC 217
522	L = 102		FUNC 218
	GO TO 260		FUNC 219
C	SINH		FUNC 220
529	ASSIGN 530 TO INDEX		FUNC 221
530	Y = FEXP(X)		FUNC 222
	IF(Y.EQ.0.) GO TO 265		FUNC 223
	X = .5 * (Y + 1. / Y) * TANH(X)		FUNC 224
	GO TO 275		FUNC 225
C	COSH		FUNC 226
539	ASSIGN 540 TO INDEX		FUNC 227
540	Y = FEXP(X)		FUNC 228
	IF(Y.EQ.0.) GO TO 265		FUNC 229
	X = .5 * (Y + 1. / Y)		FUNC 230
	GO TO 275		FUNC 231
C	TANH		FUNC 232
549	ASSIGN 550 TO INDEX		FUNC 233
550	X = TANH(X)		FUNC 234
	GO TO 275		FUNC 235
C	COTH		FUNC 236
559	ASSIGN 560 TO INDEX		FUNC 237
560	IF(X.EQ.0.) X=1.E-75		FUNC 238
	X = 1. / TANH(X)		FUNC 239
	GO TO 275		FUNC 240
C	ASINH		FUNC 241
569	ASSIGN 570 TO INDEX		FUNC 242
570	IF(ABS(X).LT.1.E-7) GO TO 265		FUNC 243
	X = SIGN(ALOG(ABS(X) + SQRT(X ** 2 + 1.)), X)		FUNC 244
	GO TO 275		FUNC 245
C	ACOSH		FUNC 246
579	ASSIGN 580 TO INDEX		FUNC 247
580	IF(X.LT.1.) GO TO 349		FUNC 248
	IF(X.GT.1.E5) GO TO 585		FUNC 249
	X=ALOG(X+SQRT(X**2-1.))		FUNC 250
	GO TO 275		FUNC 251
585	X=ALOG(2.*X)		FUNC 252
	GO TO 275		FUNC 253
C	ATANH		FUNC 254
589	ASSIGN 590 TO INDEX		FUNC 255
590	IF(ABS(X) .LT. 1.) GO TO 592		FUNC 256
	L = 103		FUNC 257
	GO TO 230		FUNC 258
592	X = .5 * ALOG((1. + X) / (1. - X))		FUNC 259
	GO TO 275		FUNC 260
C	ACOTH		FUNC 261
599	ASSIGN 600 TO INDEX		FUNC 262
600	IF(ABS(X) .LE. 1.) GO TO 602		FUNC 263
	X = .5 * ALOG((X + 1.) / (X - 1.))		FUNC 264
	GO TO 275		FUNC 265
602	L = 103		FUNC 266
	GO TO 260		FUNC 267
609	ASSIGN 610 TO INDEX		FUNC 268
610	IF(X.GT.1.OR.X.LT.0.) GO TO 349 .		FUNC 269
			FUNC 270

X0=X
X=YORHP(X0)
GO TO 275
END

FUNC 271
FUNC 272
FUNC 273
FUNC 274

SUBROUTINE GENER	GENE 1
COMMON / BLOCKA/MODE,M,KARD(77),KARO,ARO,AR02,NENCO(19),KROEND	GENE 2
1,NENCO(19,51),KSAVE,NSAVE,NFLAO	GENE 3
COMMON / BLOCKD / RC(2493),IAROS(69),KIND(39),AROTAB(51),NRHAX,	GENE 4
1 NRON,NCOL,NAROS,VWXYZ(5)	GENE 5
DIMENSION AROS(1)	GENE 6
EQUIVALENCE(IAROS(1),RC(24011))	GENE 7
C THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMAND GENERATE.	GENE 8
IF(NAROS .GE. 4 .AND. MOD(NAROS, 2) .EQ. 0) GO TO 20	GENE 9
CALL ERROR (10)	GENE 10
GO TO 200	GENE 11
C GET STORAGE COLUMN ADDRESS	GENE 12
20 CALL ADDRESS(NAROS, J)	GENE 13
IF(J .GT. 0) GO TO 30	GENE 14
3 CALL ERROR(3)	GENE 15
GO TO 200	GENE 16
C CONVERT INTEGERS TO FLOATING POINT	GENE 17
30 DO 40 I = 2, NAROS	GENE 18
IF(KIND(I-1) .EQ. 0) AROS(I-1) = IAROS(I-1)	GENE 19
40 CONTINUE	GENE 20
K=0	GENE 21
DO 50 I=4,NAROS,2	GENE 22
A=(AROS(I-1)-AROS(I-3))/AROS(I-2)	GENE 23
IF(A.LT.0.) GO TO 3	GENE 24
50 K=K+IFIX(A+.99)+1	GENE 25
IF(K.LE.NRON) GO TO 60	GENE 26
CALL ERROR(201)	GENE 27
GO TO 60	GENE 28
60 CALL PL0K	GENE 29
65 IF(NFLAG.EQ.1) RETURN	GENE 30
RC(J) = AROS(1)	GENE 31
NORON = J + NRON - 1	GENE 32
DO 150 I = 4, NAROS, 2	GENE 33
B = SIGN(1., AROS(I - 2))	GENE 34
ENDER = AROS(I - 1) - .01 * AROS(I - 2)	GENE 35
100 J = J + 1	GENE 36
RC(J) = RC(J - 1) + AROS(I - 2)	GENE 37
IF(B * (RC(J) - ENDER) .GE. 0.) GO TO 120	GENE 38
110 IF(J .LT. NORON) GO TO 100	GENE 39
GO TO 150	GENE 40
C PASSES GENERATE UPPER BOUND, SET IN UPPER BOUND	GENE 41
120 RC(J) = AROS(I - 1)	GENE 42
130 CONTINUE	GENE 43
150 NRHAX = MAX(NRHAX, J - NORON + NRON)	GENE 44
200 RETURN	GENE 45
END	GENE 46
	GENE 47

SUBROUTINE INPUT
 COMMON / BLOCKA/HODE,M,KARD(77),KARO,ARG,ARO2,NEWCO(19),KROEND
 1,NEHCDS(10.5),KSAVE,NSAVE,NFLAG

C
C
C

THIS SUBROUTINE READS IN THE LINES FROM THE REPLY AREA.

NC = 76
 CALL GRPLY(NEWCO,NC)
 KARD(1)=0
 KARD(2)=0
 KARD(KROEND+9) = 40
 CALL OMCONV(NEWCO, KARD(9), KROEND)
 RETURN
 END

INPU 1
 INPU 2
 INPU 3
 INPU 4
 INPU 5
 INPU 6
 INPU 7
 INPU 8
 INPU 9
 INPU 10
 INPU 11
 INPU 12
 INPU 13
 INPU 14

	SUBROUTINE INVCHK(NB,DET,JP)	INVC 1
	COMMON/BLOCKE/NAME(4),L1,L2,16RFLO	INVC 2
	COMMON / BLOCKO / RC(2499),IARGO(69),KIND(99),AROTAB(51),NRMAX,	INVC 3
	1 NROH,NCOL,NARGO,VNXYZ(5)	INVC 4
	COMMON/SCRAT/D(80)	INVC 5
	REAL*8 A(40),ZERO,ONE,DET	INVC 6
	EQUIVALENCE (A,D)	INVC 7
		INVC 8
C	THIS SUBROUTINE PREPARES A MATRIX FOR INVERSION BY MOVING IT TO A	INVC 9
C	SCRATCH AREA. IT ALSO CHECKS THE INVERTED MATRIX FOR ACCURACY	INVC 10
C	USING ERR TO STORE THREE MEASURES OF ACCURACY.	INVC 11
C		INVC 12
C	N1 WILL CONTAIN THE DIMENSION OF THE MATRIX TO BE INVERTED.	INVC 13
C		INVC 14
C	DET=0 IF MATRIX IS SINGULAR.	INVC 15
C	JC IS USED WHEN A SYSTEM OF LINEAR EQUATIONS IS TO BE SOLVED.	INVC 16
C	IT INDICATES WHERE THE Y VECTOR IS LOCATED.	INVC 17
C		INVC 18
	DATA ZERO/0.00/,ONE/1.00/	INVC 19
	NA=IARGO(3)	INVC 20
	DET=ZERO	INVC 21
	IF(L2.EQ.2) JC=JP	INVC 22
	NBD=ND+1	INVC 23
	NDE=2+NB	INVC 24
	JAP=IARGO(1)	INVC 25
	DO 10 I=1,NA	INVC 26
	JA=JAP	INVC 27
	DO 9 J=1,NA	INVC 28
	A(J)=ZERO	INVC 29
	A(J)=RC(JA)	INVC 30
9	JA=JA+NROH	INVC 31
	IF(L2.EQ.1) GO TO 11	INVC 32
	A(NB)=RC(JC)	INVC 33
	JC=JC+1	INVC 34
11	DO 12 J=NBD,NDE	INVC 35
	A(J)=ZERO	INVC 36
	IF(J-NB.EQ.1) A(J)=ONE	INVC 37
12	CONTINUE	INVC 38
	CALL OCRAN(1,2)	INVC 39
10	JAP=JAP+1	INVC 40
	IF(L2.EQ.1) GO TO 14	INVC 41
	DO 13 J=1,NDE	INVC 42
13	A(J)=ZERO	INVC 43
	A(NB)=-ONE	INVC 44
	A(NDE)=ONE	INVC 45
	CALL SCRAN(NB,2)	INVC 46
14	CALL GPINV(NB,DET)	INVC 47
	RETURN	INVC 48
	END	INVC 49

```

SUBROUTINE INVERT
COMMON / BLOCKA/MODE,M,KARO(77),KARO,ARO,ARO2,NEWCO(19),KROEND
1.NEHCAS(19,5),KSAVE,NSAVE,NFLAG
COMMON/SCAT/B(50)
COMMON/BLOCKE/NAKE(4),L1,L2,ICRFLO
COMMON / BLOCKD / KC(2499),IAROS(63),KIND(39),AROTAB(51),NRMAX,
1 NROW,NCOL,NAROS,VWXYZ(5)
DIMENSION TEXT(18)
REAL*8 A(40),DET
EQUIVALENCE (A,B)
C===== THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS MINVERT.
C===== INVERT, NLINEAR, AND LINEAR.
C===== L2=1 - INVERT, MINVERT
C===== L2=2 - LINEAR, NLINEAR
IF(NAROS.EQ.6.OR.NAROS.EQ.5) GO TO 1200
CALL ERROR(10)
RETURN
1200 J=NAROS
CALL CKIND (J)
IF(J.NE.0.OR.IAROS(3).NE.IAROS(4).AND.NAROS.EQ.6) GO TO 200
IF(NAROS.EQ.6) GO TO 00
KIND(6)=0
IAROS(6)=IAROS(5)
IAROS(5)=IAROS(4)
IAROS(4)=IAROS(3)
90 J=1
IF(L2.EQ.2) GO TO 95
J=2
IAROS(8)=IAROS(4)
IAROS(7)=IAROS(3)
95 CALL MTXCHK(J)
IF(J-1) 96,200,205
96 IF(IAROS(3).GT.15) GO TO 230
H1=IAROS(3)
IF(L2.EQ.1) GO TO 99
H1=H1+1
CALL ADDRESS(5,JC)
CALL ADDRESS(6,J)
IF(J.LE.0.OR.JC.LE.0) GO TO 211
99 CALL PLEX
IF(NFLAG.EQ.1) RETURN
CALL INVCHK(H1,DET,JC)
C===== CHECK TO SEE IF MATRIX HAS INVERTED. NO, IF DET=0.
IF(DET.EQ.0.00) GO TO 240
IA=IAROS(3)
JE=24111
IF(L2.EQ.2) GO TO 130
C===== STORE INVERTED MATRIX
JB=IAROS(5)
JD=H1+1
DO 110 I=1,IA
CALL SCRAM(I,1)
JC=JD
DO 100 J=JD,JE

```

```

INVE 1
INVE 2
INVE 3
INVE 4
INVE 5
INVE 6
INVE 7
INVE 8
INVE 9
INVE 10
INVE 11
INVE 12
INVE 13
INVE 14
INVE 15
INVE 16
INVE 17
INVE 18
INVE 19
INVE 20
INVE 21
INVE 22
INVE 23
INVE 24
INVE 25
INVE 26
INVE 27
INVE 28
INVE 29
INVE 30
INVE 31
INVE 32
INVE 33
INVE 34
INVE 35
INVE 36
INVE 37
INVE 38
INVE 39
INVE 40
INVE 41
INVE 42
INVE 43
INVE 44
INVE 45
INVE 46
INVE 47
INVE 48
INVE 49
INVE 50
INVE 51
INVE 52
INVE 53
INVE 54

```

RC(JC)=A(J)	INVE 55
100 JC=JC+NRCH	INVE 56
110 JB=JD+1	INVE 57
GO TO 180	INVE 58
#### STORE RESULTS OF SOLUTION	INVE 59
130 DO 140 I=1,IA	INVE 60
CALL SCRAN(I,1)	INVE 61
RC(J)=A(JE)	INVE 62
140 J=J+1	INVE 63
150 NDUN=4	INVE 64
WRITE(NDUN,100) DET	INVE 65
160 FORMAT('THE DETERMINANT OF THE INVERTED MATRIX IS',1P019.8)	INVE 66
CALL FETCH(TEXT,JD,41000)	INVE 67
CALL OSKSP(3)	INVE 68
CALL CROPLY(TEXT,JD,41000)	INVE 69
CALL CROPLY(' ',1,41000)	INVE 70
CALL CROPLY(' ',1,41000)	INVE 71
RETURN	INVE 72
200 CALL ERROR(3)	INVE 73
RETURN	INVE 74
205 CALL ERROR(17)	INVE 75
RETURN	INVE 76
211 CALL ERROR(11)	INVE 77
RETURN	INVE 78
290 CALL ERROR(23)	INVE 79
#### PRINT MATRIX TOO LARGE TO INVERT	INVE 80
RETURN	INVE 81
240 CALL ERROR(100)	INVE 82
#### PRINT MATRIX IS SINGULAR OR NEAR SINGULAR-NO INVERSE	INVE 83
1000 RETURN	INVE 84
END	INVE 85

	SUBROUTINE LOOKUP	LOOK 1
	COMMON/BLOCKE/NAME(4),L1,L2,ISRFLO	LOOK 2
	DIMENSION IR(16),MX(7),H(12),JF(78),IO(4),MA(20),MH(12),	LOOK 3
	MB(24),JT(10),JZ(30),MV(12),MJ(22),IO(36)	LOOK 4
C		LOOK 5
C	THIS SUBROUTINE IS USED TO SEARCH FOR A KEY WORD CORRESPONDING TO	LOOK 6
C	THE ONE IN THE REPLY AREA. IF ONE IS FOUND, TWO VARIABLES,	LOOK 7
C	L1 AND L2, ARE SET WHICH WILL BE USED IN DETERMINING WHICH	LOOK 8
C	SUBROUTINE TO CALL LATER TO EXECUTE A SPECIFIC COMMAND.	LOOK 9
C		LOOK 10
C	NRMAX,V,M,X,Y,Z/	LOOK 11
C		LOOK 12
C	DATA N	LOOK 13
C	1/16039,16767,17496,18225,18984,19705,5=0,1377/	LOOK 14
C		LOOK 15
C	ADS,EXP,LOG,SQRT,NEGEXP,LOOTEN,ANTILOG,SINH,COSH,TANH,COTH,ASINH,	LOOK 16
C	ACOSH,ATANH,ACOTH,DEVHOR,ABSOLUTE,EXPONENT,LOGE	LOOK 17
C	SIN,COS,TAN,COT/ARCSIN,ARCCOS,ARCTAN,ARCCOT/GINO,COSD,TAND,COTD/	LOOK 18
C	ASIND,ACOSD,ATAND,ACOTD/ASIN,ACOS,ATAN,ACOT/	LOOK 19
C		LOOK 20
C	DATA JF/002,0,4309,0,9100,0,14329,14500,10349,1309,9100,14729,1127/	LOOK 21
C	1,0900,14100,5032,2011,5032,14621,5032,2612,5032,1251,10422,025,140/	LOOK 22
C	267,1270,10422,025,14790,3073,10620,602,11290,4909,11310,9100,3015,LOOK	LOOK 23
C	314100,0,2011,0,14621,0,2012,0,1210,14100,1219,2011,1210,14621,1210/	LOOK 24
C	4,2012,14100,2910,2011,2910,14621,2910,2012,2910,1251,10314,025,	LOOK 25
C	9 19959,	LOOK 26
C	51270,10314,025,14000,1251,10200,025,19951,1270,10206,025,14500/	LOOK 27
C		LOOK 28
C	ADD,SUB,MULT,DIV,RAISE,SQRTA,MULTIPLY,DIVIDE/	LOOK 29
C		LOOK 30
C	DATA IR(1),IR(2),IR(3),IR(4),IR(5),IR(6),IR(7),IR(8),IR(9),IR(10),LOOK	LOOK 31
C	1 IR(11),IR(12),IR(13),IR(14),IR(15),IR(16)/311,0,14129,0,10056,	LOOK 32
C	2 14800,3101,0,13100,13300,14420,15007,10059,14039,3101,6674/	LOOK 33
C		LOOK 34
C	GENERATE,SET/	LOOK 35
C		LOOK 36
C	DATA IO/5252,4132,14000,0/	LOOK 37
C		LOOK 38
C	HDEFINE,ADDFINE,ADTAD,ADTAD,AREZERO,AREZERO,AREZERO,AREZERO,AREZERO,	LOOK 39
C	HYTRACE/	LOOK 40
C		LOOK 41
C	DATA HN / 9590, 4631, 042, 4031,	LOOK 42
C	4 040, 910,8091, 010, 10131, 13527, 1400, 13527, 9630, 1247,	LOOK 43
C	9 032, 1247, 9724, 4043, 10030, 015/	LOOK 44
C		LOOK 45
C	HINVERT,LINEAR,INVERT,LINEAR	LOOK 46
C	MULT,RAISE/	LOOK 47
C		LOOK 48
C	DATA HN/9734,10191,3003,3030,6001,4151,9010,10942,	LOOK 49
C	1 8049,0200,0504,7079/	LOOK 50
C		LOOK 51
C	HAAD,AGUA,HTRANS,AGAD,AGUA,MULT,ADIVIDE,RAISE,CCALAR,ATRAUS,	LOOK 52
C	ASCALAR,ASCALAR	LOOK 53
C		LOOK 54

```

DATA MB(1),MB(2),MB(3),MB(4),MB(5),MB(6),MB(7),MB(8) / LOOK 55
1 9500, 2916, 10011, 1450, 10095, 1126, 760, 2916 / LOOK 56
DATA MB(9),MB(10),MB(11),MB(12),MB(13),MB(14),MB(15),MB(16) / LOOK 57
1 1269, 1450, 1101, 9200, 846, 16205, 1216, 7079 / LOOK 58
DATA MB(17), MB(18), MB(19), MB(20), MB(21), MB(22), MB(23), MB(24) / LOOK 59
113933, 0793, 1297, 1125, 1245, 1054, 9393, 1054 / LOOK 60
C LOOK 61
C PARSUM,PARPROD,RMS,AVERAGE,SUM LOOK 62
C LOOK 63
DATA JT(1),JT(2),JT(3),JT(4),JT(5),JT(6),JT(7),JT(8),JT(9),JT(10) / LOOK 64
1 11709, 14431, 11709, 12105, 13402, 0, 1320, 19158, 14431, 0 / LOOK 65
C LOOK 66
C RORCUM,PRODUCT,DEFINE,MAX,MAXIMUM,MIN,MINIMUM, SORT,ORDER, LOOK 67
C ERAGE,EXCHANGE,FLIP,CHANGE,HIERRARCHY LOOK 68
C LOOK 69
DATA JZ / 13850,14491,12105,3485,3057,6944,9520,0,9520,6903,9734,0, LOOK 70
1 9734,6903,14274,14500,11425,4101,4132,10906,4200,5075,4707,11604, LOOK 71
2 2404,10400,6000,19167,10705,8742 / LOOK 72
C LOOK 73
C CLOSE,COUNT,SHORTEN,EXPAND,DUPLICATE,MOVE,BLOCKTRANSFER,MOVE, LOOK 74
C MOVE,PROKOTE,DENOTE LOOK 75
C LOOK 76
DATA HJ(1),HJ(2),HJ(3),HJ(4),HJ(5),HJ(6),HJ(7),HJ(8),HJ(9),HJ(10), LOOK 77
1 HJ(11),HJ(12),HJ(13),HJ(14),HJ(15),HJ(16),HJ(17),HJ(18),HJ(19), LOOK 78
2 HJ(20),HJ(21),HJ(22) / 2826,10000,2019,10743,14002, LOOK 79
4 19387,4909,1111,3489,0034,0004,3613,1797,2504, LOOK 80
5 1065,10173,9943,10173,12100,5902,3034, LOOK 81
6 11460 / LOOK 82
C LOOK 83
C XX,X,XX,AD,CA,RV,V LOOK 84
C LOOK 85
DATA MX/10144,17493,17047,037,2949,1929,16030 / LOOK 86
C LOOK 87
C MYECODING,AVECODING,MVACHAT,AVECARR,MNATVSC,ANARVEC LOOK 88
C LOOK 89
DATA MY / 10070,2304,1020,2304, LOOK 90
2 10070,2304,1020,2304,0329,10173,774,13721 / LOOK 91
C LOOK 92
C YORAX,YORAP,YORAZ,ORAX,ORAP,ORAZ,CHIX,CHIP,CHIZ LOOK 93
C YTX,YTP,YTZ,DETAX,DETAP,DETAE,FFX,FFP,FFZ LOOK 94
C LOOK 95
DATA IZ/10045,10125,10049,0709,10040,10179,5143,17400,5143,11004, LOOK 96
1 5143,10125,2412,17400,0412,11004,2412,10031,10144,0,13108,0, LOOK 97
2 10146,0,1013,1077,1013,1101,1013,1491,4000,0,4002,0,4002,0 / LOOK 98
C LOOK 99
C CHECK NAMES WITH CURLIFIERS FIRST LOOK 100
C LOOK 101
C LOOK 102
C LOOK 103
C REGCT: NAMEX,V,N,X,Y,Z L1 = 1, L2 = 1 - 6 LOOK 104
C LOOK 105
C IF(NAME(1),NE,13270-03,NAME(2),NE,4103100 TO 140 LOOK 106
C DO 104 L2=1,6 LOOK 107
C IF(NAME(3),EQ,NAME(2),AND,NAME(4),EQ,NAME(2+6)) GO TO 106 LOOK 108

```

104	CONTINUE	LOOK 109
	GO TO 899	LOOK 110
108	L1=1	LOOK 111
	GO TO 900	LOOK 112
C		LOOK 113
C	READ L1 = 2	LOOK 114
C		LOOK 115
140	IF(NAME(1).NE.19250.OR.NAME(2).NE.2916) GO TO 170	LOOK 116
	L1 = 2	LOOK 117
	GO TO 900	LOOK 118
C		LOOK 119
C	M(X'X'),M(X'X'),M(X'AX'),M(XAX'),M(AD),M(DA),M(RV),M(V'A)	LOOK 120
C	L1 = 3, L2 = 1 - 7	LOOK 121
C		LOOK 122
170	IF(NAME: 1) .NE. 9477) GO TO 180	LOOK 123
	L1 = 3	LOOK 124
	DO 174 L2 = 1, 7	LOOK 125
	IF(NAME(3) .EQ. MX(L2)) GO TO 900	LOOK 126
174	CONTINUE	LOOK 127
	GO TO 899	LOOK 128
C		LOOK 129
C	MVECDIAG,AVECDIAG,MVECMAT,AVECARR,MHATVEC,ARRRVEC L1=13, L2=1-6	LOOK 130
C		LOOK 131
180	DO 184 L2=1,6	LOOK 132
	IF(NAME(1).EQ.HV(2*L2-1).AND.NAME(2).EQ.HV(2*L2)) GO TO 180	LOOK 133
184	CONTINUE	LOOK 134
	GO TO 200	LOOK 135
186	L1=19	LOOK 136
	GO TO 900	LOOK 137
C	ADD,SUB,MULT,DIV,RAISE,SUBTRA,MULTIPLY,DIVIDE L1 = 4, L2 = 1 - 8	LOOK 138
C		LOOK 139
C		LOOK 140
200	DO 204 L2=2,10,2	LOOK 141
	IF(NAME(1).EQ.IR(L2-1).AND.NAME(2).EQ.IR(L2)) GO TO 208	LOOK 142
204	CONTINUE	LOOK 143
	GO TO 210	LOOK 144
208	L1 = 4	LOOK 145
	L2=L2/2	LOOK 146
	GO TO 900	LOOK 147
C		LOOK 148
C		LOOK 149
C	ABS,EXP,LOG,SQRT,NEOEXP,LOGTEN,ANTILOG,SINH,COSH,TANH,COTH,ASINH,	LOOK 150
C	ACOSH,ATANH,ACOTH,DEVHQR,ABSOLUTE,EXPONENT,LOG,	LOOK 151
C	SIN,COS,TAN,COT,ARCSIN,ARCCOS,ARCTAN,ARCCOT,SIND,COSD,TAND,COTD,	LOOK 152
C	ASIND,ACOSD,ATAND,ACOTD,ASIN,ACOS,ATAN,ACOT, L1=5,L2=1,39	LOOK 153
C		LOOK 154
210	L1=5	LOOK 155
	DO 224 L2=2,70,2	LOOK 156
	IF(NAME(1).EQ.JF(L2-1).AND.NAME(2).EQ.JF(L2)) GO TO 228	LOOK 157
224	CONTINUE	LOOK 158
	GO TO 230	LOOK 159
226	L2=L2/2	LOOK 160
	GO TO 900	LOOK 161
C		LOOK 162

C	GENERATE.SET	L1 = 0. L2 = 1,2	LOOK 183
C			LOOK 184
230	DO 234 L2 = 2, 4,2		LOOK 185
	IF(NAME(1).EQ.JO(L2-1).AND.NAME(2).EQ.JO(L2)) GO TO 236		LOOK 186
234	CONTINUE		LOOK 187
	GO TO 250		LOOK 188
236	L1 = 6		LOOK 189
	L2 = L2 / 2		LOOK 170
	GO TO 300		LOOK 171
C			LOOK 172
C	NODEFINE,ACDEFINE,ADIAO,MDIAO,MZERO,AZERO,HERAGE,AERASE,MIDLAT		LOOK 173
C	MTRACE/		LOOK 174
C	L1 = 7. L2 = 1 - 10		LOOK 175
C			LOOK 176
250	DO 254 L2 = 1, 10		LOOK 177
	IF(NAME(1).EQ.MA(2*L2-1).AND.NAME(2).EQ.MA(2*L2)) GO TO 256		LOOK 178
254	CONTINUE		LOOK 179
	GO TO 260		LOOK 180
256	L1 = 7		LOOK 181
	GO TO 300		LOOK 182
C			LOOK 183
C	MINVERT,LINEAR,INVERT,MLINER	L1 = 0. L2 = 1, 2	LOOK 184
C	MMULT,MRAISE	L1 = 0. L2 = 1, 2	LOOK 185
C			LOOK 186
260	DO 264 L2 = 1, 0		LOOK 187
	IF(NAME(1).EQ.MM(2*L2-1).AND.NAME(2).EQ.MM(2*L2)) GO TO 266		LOOK 188
264	CONTINUE		LOOK 189
	GO TO 270		LOOK 190
266	L1 = 0		LOOK 191
	IF(L2.GT.4) L1=9		LOOK 192
	L2=MOD(L2+1,2)+1		LOOK 193
	GO TO 300		LOOK 194
C			LOOK 195
C	MADD,MSUB,MTRANS,MADD,ASUB,AMULT,ADIVIDE,MRAISE,SCALAR,ATRANS,		LOOK 196
C	MSCALAR	L1 = 10, L2 = 1 - 9	LOOK 197
C			LOOK 198
270	DO 274 L2 = 1, 12		LOOK 199
	IF(NAME(1).EQ.MB(2*L2-1).AND.NAME(2).EQ.MB(2*L2)) GO TO 276		LOOK 200
274	CONTINUE		LOOK 201
	GO TO 290		LOOK 202
276	L1 = 10		LOOK 203
	IF(L2 -10) 300, 277, 278		LOOK 204
277	L2 = 3		LOOK 205
	GO TO 300		LOOK 206
278	L2 = 9		LOOK 207
	GO TO 300		LOOK 208
C			LOOK 209
C	PARSUB,PARPROD,MMS,AVERAGE,GUM	L1 = 11, L2 = 1 - 6	LOOK 210
C			LOOK 211
290	L1=11		LOOK 212
	DO 294 L2 = 1, 6		LOOK 213
	IF(NAME(1).EQ.JT(2*L2-1).AND.NAME(2).EQ.JT(2*L2)) GO TO 300		LOOK 214
294	CONTINUE		LOOK 215
C			LOOK 216

C	RONGUN.PRODUCT.DEFINE.MAX.MAXIMUM.MIN.MINIMUM.SORT.ORDER.	LOOK 217
C	ERASE.EXCHANGE.FLIP.CHANGE.HIERARCHY L1 = 12 L2 = 1 - 14	LOOK 218
C		LOOK 219
	L1 = 12	LOOK 220
	DO 904 L2=1.15	LOOK 221
	IF(NAME(1).EQ. JZ(2=L2-1).AND. NAME(2).EQ. JZ(2=L2)) GO TO 900	LOOK 222
904	CONTINUE	LOOK 223
C		LOOK 224
C	CLOSE.COUNT.SHORTEN.EXPAND.DUPLICATE.MOVE.BLOCKTRANSFER.AMOVE.	LOOK 225
C	MMOVE.PROMOTE.DENOTE L1 = 14. L2 = 1 - 11	LOOK 226
C		LOOK 227
	L1 = 14	LOOK 228
	DO 924 L2 = 1. 11	LOOK 229
	IF(NAME(1).EQ.H 2=L2-1).AND.NAME(2).EQ.HJ(2=L2)) GO TO 900	LOOK 230
924	CONTINUE	LOOK 231
C		LOOK 232
C	TTX.TTP.TTZ.BETAX.DETAP.BETAZ.FFX.FFP.FFZ L1=12.L2=1-14	LOOK 233
C	YORHX.YORHP.YORHZ.GRHX.GRHP.GRHZ.CHIX.CHIP.CHIZ	LOOK 234
C		LOOK 235
	L1=15	LOOK 236
	DO 900 L2=1.10	LOOK 237
	IF(NAME(1).EQ.I0(2=L2-1).AND.NAME(2).EQ.I0(2=L2)) GO TO 900	LOOK 238
900	CONTINUE	LOOK 239
099	L1=0	LOOK 240
900	RETURN	LOOK 241
	END	LOOK 242

MAIN AND CROSS REFERENCE TABLE		MAIN	1
		MAIN	2
		MAIN	3
THIS IS A CROSS-REFERENCE TABLE SHOWING WHICH SUBPROGRAMS		MAIN	4
REFERENCE PARTICULAR BLOCKS OF COMMON OR PARTICULAR SUBPROGRAMS.		MAIN	5
THIS LIST DOES NOT INCLUDE THE FOLLOWING SUBROUTINES WHICH ARE		MAIN	6
CALLED ONLY BY THE SUBROUTINE EXECUTE.		MAIN	7
ARITH CHANGE DEFINE ERASE EXCHNG EXPCON EXTREN FLIP		MAIN	8
FUNCT GENER INVERT MATRIX MISC2 MMULT MOP MOVE		MAIN	9
MRAISE MSCRON MXTX PDMOTE PRORON READX RESET SET		MAIN	10
SORDER		MAIN	11
OMNITAS USES UNLABELLED COMMON IN ERROR. OMNIT. PLBK. PROGRAM.		MAIN	12
NORKO. EXECUTE. XOMNIT.		MAIN	13
***** LABELLED COMMON *****		MAIN	14
		MAIN	15
		MAIN	16
		MAIN	17
BLOCKA		MAIN	18
AAROS ARITH ARYVEC ASTER BLOCK CHANGE DEFINE ERASE		MAIN	19
ERROR EXCHNG EXPCON EXTREN FLIP FUNCT GENER INPUT		MAIN	20
INVERT MATRIX MDMAD MISC2 MMULT MOP MOVE MRAISE		MAIN	21
MSCRON MXTX MNAME MONGLA OMNIT PDMOTE PHYCON PLBK		MAIN	22
PROGRAM PRORON READQ READX RESET SET SETQ SORDER		MAIN	23
STAT TRANSF VARCON EXECUTE XOMNIT		MAIN	24
BLOCKB		MAIN	25
ADRESS ARITH ARYVEC BLOCK CHANGE CHNCOL CKIND DEFINE		MAIN	26
ERASE EXCHNG EXPAND EXPCON EXTREN FLIP FUNCT GENER		MAIN	27
INVCHK INVERT MATRIX MDMAD MISC2 MMULT MOP MOVE		MAIN	28
MRAISE MSCRON MIXCHK MXTX OMNIT PDMOTE PRORON READQ		MAIN	29
READX RESET SET SETQ SORDER TRANSF VECTOR NORKO		MAIN	30
XOMNIT XPND		MAIN	31
BLOCKC		MAIN	32
ARITH ARYVEC BLOCK DEFINE EXPAND EXPCON EXTREN FUNCT		MAIN	33
INVCHK INVERT LOOKUP MATRIX MDMAD MISC2 MOP MSCRON		MAIN	34
MXTX OMNIT PDMOTE PRORON READX RESET SET SORDER		MAIN	35
TRANSF EXECUTE		MAIN	36
BLOCKF		MAIN	37
ADRESS BLOCK MXTX		MAIN	38
CONSTS		MAIN	39
BLOCK FUNCT		MAIN	40
KPILOT		MAIN	41
BLOCK ERROR OMNIT PLBK PROGRAM		MAIN	42
PCONST		MAIN	43
BLOCK PHYCON		MAIN	44
QRS		MAIN	45
OMNIT READQ READX SET SETQ		MAIN	46
SCRAT		MAIN	47
ARYVEC EXPCON INVCHK INVERT MATRIX MDMAD MISC2 MMULT		MAIN	48
MOP MOVE MRAISE MXTX PRORON SORDER SPINV TRANSF		MAIN	49
		MAIN	50
		MAIN	51
		MAIN	52
***** SUBROUTINES AND FUNCTIONS *****		MAIN	53
		MAIN	54

[illegible]

FFZ	STATO	MAIN	109
FLOO		MAIN	110
FSIN	FUNCT	MAIN	111
FSQRT	FUNCT	MAIN	112
GAMP	FUNCT INVCHK MSCRON	MAIN	113
GAMP	CHIP STATO	MAIN	114
GAMX		MAIN	115
GAMZ	BETAX CHIX GAMP STATO	MAIN	116
INPUT	CHIZ GAMP STATO	MAIN	117
INVCHK	OMNIT	MAIN	118
LOOKUP	INVERT	MAIN	119
NDAMAD	OMNIT	MAIN	120
NTXCHK	NXTX	MAIN	121
	ARYVEC EXPCON INVERT MATRIX NDAMAD MISCZ MMULT MOP	MAIN	122
	MOVE NRISE NXTX TRANSF	MAIN	123
NNAME	ASTER OMNIT	MAIN	124
NONDLA	ASTER	MAIN	125
ONCONV	INPUT	MAIN	126
PHYCON	ASTER	MAIN	127
PLDK	ARITH ARYVEC CHANGE DEFINE ERASE EXCHNG EXPCON EXTREM	MAIN	128
	FLIP FUNCT CENER INVERT MATRIX NDAMAD MISCZ MMULT	MAIN	129
	NOP MOVE NRISE MSCRON NXTX PCHOTE PRORON REGET	MAIN	130
	SET SETQ SORDER TRANSF	MAIN	131
PROGRAH	ERROR OMNIT XECUTE	MAIN	132
READQ	OMNIT	MAIN	133
SCRAN	INVCHK INVERT MATRIX NDAMAD MISCZ MMULT MOVE NXTX	MAIN	134
	SPINV TRANSF	MAIN	135
SETQ	OMNIT	MAIN	136
SPINV	INVCHK	MAIN	137
TRANSF	NXTX	MAIN	138
TTP	STATO	MAIN	139
TTX		MAIN	140
		MAIN	141
		MAIN	142
		MAIN	143
		MAIN	144
		MAIN	145
		MAIN	146
		MAIN	147
		MAIN	148
		MAIN	149
		MAIN	150
		MAIN	151
		MAIN	152
		MAIN	153
		MAIN	154
		MAIN	155
		MAIN	156
		MAIN	157
		MAIN	158
		MAIN	159
		MAIN	160
		MAIN	161
		MAIN	162

C C	STATO	MAIN 169
	ITZ	MAIN 164
	STATO	MAIN 165
	VARCON	MAIN 166
	ASTER	MAIN 167
	VECTOR	MAIN 168
	DEFINE ERAGE EXTREM FUNCT MISC2 MSCROW PDMOTE	MAIN 169
	WORKO	MAIN 170
	OMNIT	MAIN 171
	XECUTE	MAIN 172
	OMNIT	MAIN 173
	XOMNIT	MAIN 174
	OMNIT	MAIN 175
	XPND	MAIN 176
	EXPAND	MAIN 177
	YORHP	MAIN 178
	CAHP FUNCT STATO	MAIN 179
	YORHX	MAIN 180
	MATRIX MORAND MISC2 HOP MSCROW MTXCHK SET XPND	MAIN 181
	YORHZ	MAIN 182
STATO YORHP	MAIN 183	
COMMON KEY.IOVLY.ITYPE	MAIN 184	
DEFINE FILE 26(100.96.U.IOVLY)	MAIN 185	
CALL PLOTS(I,J,O)	MAIN 186	
CALL GRINIT('D')	MAIN 187	
IOVLY=)	MAIN 188	
CALL OMNIT	MAIN 189	
CALL ORRIGE	MAIN 190	
CALL PLOT(20..0..999)	MAIN 191	
STOP	MAIN 192	
END	MAIN 193	

SUBROUTINE MATRIX	MATR	1
COMMON / BLOCKA/MODE,H,KARD(77),KARG,ARG,ARG2,NEWCO(19),KRDEND	MATR	2
1,NEWCOS(19,5),KSAVE,NSAVE,NFLAO	MATR	3
COMMON / BLOCKD / RC(2493),IARG3(69),KIND(39),AROTAB(51),NRMAX,	MATR	4
1 NR0H,NCOL,NAROS,VWXYZ(5)	MATR	5
COMMON/SCRAT/RI(80)	MATR	6
COMMON/BLOCKE/NAME(4),L1,L2,ISRFL0	MATR	7
C *****	MATR	8
C THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS MADD, MSUB,	MATR	9
C MTRANS, ATRANS, AADD, ASUB, AMULT, ADIVIDE, ARAISE, ASCALAR,	MATR	10
C MSCALAR AND SCALAR.	MATR	11
C L2=1 - MADD	MATR	12
C L2=2 - MSUB	MATR	13
C L2=3 - MTRANS, ATRANS	MATR	14
C L2=4 - AADD	MATR	15
C L2=5 - ASUB	MATR	16
C L2=6 - AMULT	MATR	17
C L2=7 - ADIVIDE	MATR	18
C L2=8 - ARAISE	MATR	19
C L2=9 - ASCALAR, MSCALAR, SCALAR	MATR	20
C *****	MATR	21
C NR0HPP=NR0H	MATR	22
C K=1	MATR	23
C NR0HP=NR0H	MATR	24
C *****	MATR	25
C CHECK TO SEE IF WE HAVE CORRECT NUMBER OF ARGUMENTS	MATR	26
C IF NOT NO FURTHER CHECKING IS DONE	MATR	27
C *****	MATR	28
C IF(L2-9)100,120,140	MATR	29
C 100 IF(NAROS.NE.8.AND.NAROS.NE.10.AND.NAROS.NE.7) GO TO 400	MATR	30
C GO TO 605	MATR	31
C 120 IF(NAROS.NE.6.AND.NAROS.NE.5) GO TO 400	MATR	32
C GO TO 605	MATR	33
C 140 IF(NAROS.LT.6.OR.NAROS.GT.10.OR.NAROS.EQ.9) GO TO 400	MATR	34
C GO TO 640	MATR	35
C 400 CALL ERROR(10)	MATR	36
C RETURN	MATR	37
C *****	MATR	38
C CHECK TO SEE IF ALL ARGUMENTS ARE INTEGERS	MATR	39
C IF NOT NO FURTHER CHECKING IS DONE	MATR	40
C *****	MATR	41
C 605 J=NAROS	MATR	42
C CALL CKIND(J)	MATR	43
C 610 IF(J.EQ.0) GO TO 600	MATR	44
C 620 CALL ERROR(3)	MATR	45
C RETURN	MATR	46
C 640 N2=NAROS-2	MATR	47
C IF(1.2.EQ.3.AND.KIND(N2).EQ.0) GO TO 620	MATR	48
C IF(NAROS.GT.7) GO TO 605	MATR	49
C *****	MATR	50
C FIND ADDRESSES OF COLUMNS	MATR	51
C *****	MATR	52
C CALL ADDRESS(N2,1BP)	MATR	53
C IF(1BP) 660,620,600	MATR	54

680 IOP=-IOP	MATR 65
K=0	MATR 56
680 NROWP=0	MATR 57
KIND(N2)=KIND(NAROS)	MATR 58
IAROS(N2)=IAROS(N2+1)	MATR 59
IAROS(N2+1)=IAROS(NAROS)	MATR 60
GO TO 605	MATR 61
800 KIND(6)=0	MATR 62
KIND(10)=0	MATR 63
IF(NAROS.EQ.6.AND.L2.LE.3.OR.NAROS.EQ.7.AND.L2.GT.3.OR.NAROS.GT.7)	MATR 64
1 GO TO 600	MATR 65
N=NAROS	MATR 66
DO 850 J=3,NAROS	MATR 67
IAROS(N+1)=IAROS(N)	MATR 68
850 N=N-1	MATR 69
900 IF(NAROS.GT.8) GO TO 1500	MATR 70
IAROS(10)=IAROS(8)	MATR 71
IAROS(9)=IAROS(7)	MATR 72
IF(L2.EQ.3) GO TO 1900	MATR 73
IAROS(8)=IAROS(4)	MATR 74
IAROS(7)=IAROS(3)	MATR 75
GO TO 1400	MATR 76
1300 IAROS(8)=IAROS(3)	MATR 76
IAROS(7)=IAROS(4)	MATR 77
NROWP=1	MATR 78
1400 IF(NAROS.GT.7.OR.L2.LT.3) GO TO 1600	MATR 79
J=2	MATR 80
GO TO 1700	MATR 81
*****	MATR 82
C CHECK TO SEE IF DIMENSIONS ARE CORRECT IF THEY ARE GIVEN	MATR 83
C IF NOT NO FURTHER CHECKING IS DONE	MATR 84
C *****	MATR 85
1500 IF(IAROS(9).NE.IAROS(7).OR.IAROS(4).NE.IAROS(8)) GO TO 620	MATR 86
1600 IAROS(12)=IAROS(4)	MATR 87
IAROS(11)=IAROS(3)	MATR 88
J=9	MATR 89
1700 CALL MTXCHK(J)	MATR 90
IF(J-1) 1800,620,1750	MATR 91
1750 CALL ERRON(17)	MATR 92
RETURN	MATR 93
1800 CALL PLEX	MATR 94
IF(NFLAG.EQ.1) RETURN	MATR 95
IF(NAROS.GT.7.OR.L2.LT.3) GO TO 1900	MATR 96
ICP=IAROS(5)	MATR 97
GO TO 2000	MATR 98
1900 IOP=IAROS(5)	MATR 99
ICP=IAROS(9)	MATR 100
2000 IIB=IAROS(7)	MATR 101
JJB=IAROS(8)	MATR 102
ASSIGN 2100 TO N	MATR 103
IAP=IAROS(1)	MATR 104
DO 3560 J=1,JJB	MATR 105
IA=IAP	MATR 106
IO=IOP	MATR 107
	MATR 108
	MATR 109
	MATR 110

DO 3540 I=1,IIB	MATR 109
GO TO N.(2120,2140,2200,2160,2175,2320,2100)	MATR 110
2100 GO TO (2110,2130,2190,2110,2130,2150,2170,2310,2150),L2	MATR 111
2110 ASSIGN 2120 TO N	MATR 112
2120 A(I)=RC(IA)+RC(IB)	MATR 113
GO TO 3500	MATR 114
2130 ASSIGN 2140 TO N	MATR 115
2140 A(I)=RC(IA)-RC(IB)	MATR 116
GO TO 3500	MATR 117
2150 ASSIGN 2160 TO N	MATR 118
2160 A(I)=RC(IA)+RC(IB)	MATR 119
GO TO 3300	MATR 120
2170 ASSIGN 2175 TO N	MATR 121
2175 IF(ABS(RC(IB))-GT-.1E-60) GO TO 2180	MATR 122
A(I)=0.	MATR 123
GO TO 3500	MATR 124
2180 A(I)=RC(IA)/RC(IB)	MATR 125
GO TO 3500	MATR 126
2190 ASSIGN 2200 TO N	MATR 127
2200 A(I)=RC(IA)	MATR 128
IA=IA+NRON	MATR 129
GO TO 3540	MATR 130
2310 ASSIGN 2320 TO N	MATR 131
2320 A(I)=PEXP2(RC(IA),RC(ID))	MATR 132
3500 IB=IB+K	MATR 133
IA=IA+1	MATR 134
3540 CONTINUE	MATR 135
ICP=ICP+NRONPP	MATR 136
IDP=IDP+NRONP	MATR 137
3560 CALL SCRAN(J,2)	MATR 138
C *****	MATR 139
C MOVE RESULTS FROM SCRATCH AREA TO WORKSHEET	MATR 140
C *****	MATR 141
DO 4060 J=1,JJB	MATR 142
CALL SCRAN(J,1)	MATR 143
IC=ICP	MATR 144
DO 4080 I=1,IIB	MATR 145
RC(IC)=A(I)	MATR 146
IC=IC+1	MATR 147
4060 CONTINUE	MATR 148
ICP=ICP+NRON	MATR 149
4080 CONTINUE	MATR 150
RETURN	MATR 151
END	MATR 152

SUBROUTINE NDAAD	NDAH	1
COMMON/DLOCKE/NAME(4),L1,L2,IGRFL0	NDAH	2
COMMON / BLOCKD / RC(2439),IARG(69),KIND(39),AROTAB(51),NRMAX,	NDAH	3
1 NRON,NCOL,NARG5,VWXYZ(5)	NDAH	4
COMMON/SCRAT/RI(60)	NDAH	5
COMMON / BLOCKA/MSDE,M,KARD(77),KARG,ARG,ARG2,NEWCD(19),KIDEND	NDAH	6
1,NEWCD5(19,5),KSAVE,NSAVE,NFLAD	NDAH	7
C *****	NDAH	8
C THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS	NDAH	9
C N(AD) AND N(OR)	NDAH	10
C L2 = 4 - N(AD)	NDAH	11
C L2 = 5 - N(OR)	NDAH	12
C *****	NDAH	13
C CHECK FOR CORRECT NUMBER OF ARGUMENTS	NDAH	14
C *****	NDAH	15
IF(NARG5.NE.7) GO TO 170	NDAH	16
C *****	NDAH	17
C CHECK TO SEE THAT ALL ARGUMENTS ARE INTEGERS	NDAH	18
C *****	NDAH	19
J=NARG5	NDAH	20
CALL CKIND(J)	NDAH	21
IF(J.NE.0) GO TO 160	NDAH	22
C *****	NDAH	23
C CHECK TO SEE IF DIMENSIONS ARE OUT OF RANGE	NDAH	24
C AND COMPUTE ADDRESS OF COLUMN	NDAH	25
C *****	NDAH	26
CALL ADDRESS(5,10P)	NDAH	27
IF(10P.GT.0) GO TO 50	NDAH	28
CALL ERROR(11)	NDAH	29
RETURN	NDAH	30
50 IARG5(5)=IARG5(6)	NDAH	31
IARG5(6)=IARG5(7)	NDAH	32
IARG5(7)=IARG5(3)	NDAH	33
IARG5(8)=IARG5(4)	NDAH	34
J=2	NDAH	35
CALL RTXCHK(J)	NDAH	36
IF(J-1) 190 , 160 , 180	NDAH	37
160 CALL ERROR(13)	NDAH	38
RETURN	NDAH	39
170 CALL ERROR(10)	NDAH	40
RETURN	NDAH	41
180 CALL ERROR(17)	NDAH	42
RETURN	NDAH	43
190 CALL FLAX	NDAH	44
IF(NFLAD.EQ.1) RETURN	NDAH	45
IP=IARG5(4)	NDAH	46
JP=IARG5(3)	NDAH	47
IF(12.EQ.4) GO TO 200	NDAH	48
I1=0	NDAH	49
I2=1	NDAH	50
GO TO 260	NDAH	51
200 I1=1	NDAH	52
I2=0	NDAH	53
260 IA=IARG5(1)	NDAH	54

```

IB=IARGST(S)
DO 340 I=1,IP
ID=IDP
DO 300 J=1,JP
A(J)=RC(ID)=RC(IA)
ID=ID+12
IA=IA+1
300 CONTINUE
IA=IA+NROW-JP
IDP=IDP+11
340 CALL SCRAM(I,2)
DO 440 I=1,IP
CALL SCRAM(I,1)
DO 400 J=1,JP
RC(ID)=A(J)
ID=ID+1
400 CONTINUE
IB=IB+NROW-JP
440 CONTINUE
RETURN
END

```

```

MOAH 55
MOAH 56
MOAH 57
MOAH 58
MOAH 59
MOAH 60
MOAH 61
MOAH 62
MOAH 63
MOAH 64
MOAH 65
MOAH 66
MOAH 67
MOAH 68
MOAH 69
MOAH 70
MOAH 71
MOAH 72
MOAH 73
MOAH 74
MOAH 75

```

	SUBROUTINE NIGC2	NIGC	1
	COMMON / BLOCKA/MODE,M,KARD(77),KARG,ARG,ARG2,NEWCO(10),KROEND	NIGC	2
	1,NEWCOS(10,5),KSAVE,NSAVE,NFLAG	NIGC	3
	COMMON / BLOCKB / RC(2499),IARG(69),KIND(30),AROTAB(51),NRMAX,	NIGC	4
	1 NRON,NCOL,NARGS,VWXYZ(5)	NIGC	5
	COMMON/BLOCKC/NAME(4),L1,L2,ISRFLG	NIGC	6
	DIMENSION ARG(1)	NIGC	7
	EQUIVALENCE(ARG(1),RC(2401))	NIGC	8
	COMMON/SCRAT/AT(50)	NIGC	8
C		NIGC	10
C	THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS	NIGC	11
C	CLOSE, COUNT, SHORTEN, EXPAND AND DUPLICATE.	NIGC	12
C	L2=1 - CLOSE	NIGC	13
C	L2=2 - COUNT	NIGC	14
C	L2=3 - SHORTEN	NIGC	15
C	L2=4 - EXPAND	NIGC	16
C	L2=5 - DUPLICATE	NIGC	17
C		NIGC	18
	J=NARGS	NIGC	19
	IF(NARGS.GE.2) GO TO 40	NIGC	20
10	M = 10	NIGC	21
20	CALL ERROR(K)	NIGC	22
30	RETURN	NIGC	23
40	DO TO (50,74,50,400,600) , L2	NIGC	24
50	IF(KIND(L2).EQ.1) GO TO 70	NIGC	25
60	K = 9	NIGC	26
	DO TO 20	NIGC	27
70	KIND(L2) = 0	NIGC	28
	IF(L2.EQ.3.AND.NARGS.NE.5) GO TO 10	NIGC	29
	ARG1 = ARG(L2)	NIGC	30
	IARG(L2) = IARG(L2+1)	NIGC	31
	GO TO 75	NIGC	32
74	IF(NARGS.NE.2) GO TO 10	NIGC	33
75	CALL CHKCOL(J)	NIGC	34
	IF(J.EQ.1) GO TO 60	NIGC	35
	DO DO I=1,NARGS	NIGC	36
80	IARG(I) = IARG(I) - 1	NIGC	37
	IF(NRMAX.GT.0) GO TO 130	NIGC	38
120	M = 0	NIGC	39
	GO TO 20	NIGC	40
130	CALL PLON	NIGC	41
	IF(NFLAG.EQ.1) RETURN	NIGC	42
	IF (L2 = 2) 140,200,300	NIGC	43
C		NIGC	44
C	CLOSE	NIGC	45
C		NIGC	46
140	DO 100 J=2,NARGS	NIGC	47
	M = IARG(J)	NIGC	48
	M = 0	NIGC	49
	DO 100 I=1,NRMAX	NIGC	50
	J1 = M + 1	NIGC	51
140	IF(RC(J1).NE.ARG1) GO TO 100	NIGC	52
	M = M + 1	NIGC	53
	IF ((M+1) .EQ. (NRMAX+1)) GO TO 101	NIGC	54

K1 = J1 + 1	HISC 55
K3 = K + NRMAX	HISC 56
DO 155 K2 = K1, K3	HISC 57
155 RC(K2 - 1) = RC(K2)	HISC 58
GO TO 146	HISC 59
160 CONTINUE	HISC 60
161 IF (M .EQ. 0) GO TO 190	HISC 61
M = NRMAX - M + 1	HISC 62
DO 160 I = M, NRMAX	HISC 63
J1 = K + 1	HISC 64
180 RC(J1) = 0.0	HISC 65
190 CONTINUE	HISC 66
GO TO 30	HISC 67
C	HISC 68
C COUNT	HISC 69
C	HISC 70
200 J=IAROS(1)+NRMAX+1	HISC 71
DO 250 I=1, NRMAX	HISC 72
JJ=J-I	HISC 73
IF(RC(JJ) .NE. 0.) GO TO 260	HISC 74
250 CONTINUE	HISC 75
260 ARC1 = JJ - IAROS(1)	HISC 76
IAROS(2) = IAROS(2) + 1	HISC 77
CALL VECTOR (ARC1, IAROS(2))	HISC 78
GO TO 30	HISC 79
C	HISC 80
C SHORTEN	HISC 81
C	HISC 82
300 IF(NRMAX.EQ.1) GO TO 370	HISC 83
DO 360 K=2, NRMAX	HISC 84
J1 = IAROS(2) + K	HISC 85
IF (ARC1 - RC(J1-1)) 320, 330, 340	HISC 86
320 IF(ARC1.LT.RC(J1)) GO TO 360	HISC 87
350 NRMAX = K	HISC 88
GO TO 370	HISC 89
350 NRMAX = K - 1	HISC 90
GO TO 370	HISC 91
340 IF(ARC1.LE.RC(J1)) GO TO 360	HISC 92
360 CONTINUE	HISC 93
370 DO 380 I=1, NRMAX	HISC 94
K = IAROS(1) + 1	HISC 95
J = IAROS(4) + 1	HISC 96
M = IAROS(5) + 1	HISC 97
K1 = IAROS(2) + 1	HISC 98
RC(K) = RC(K1)	HISC 99
380 RC(J) = RC(K)	HISC 100
GO TO 30	HISC 101
C	HISC 102
C EXPAND	HISC 103
C	HISC 104
400 IF(NRROS.NE.4) GO TO 10	HISC 105
IF(KIND(2).EQ.0) ARCS(2)=IAROS(2)	HISC 106
IF(KIND(3).EQ.0) ARCS(3)=IAROS(3)	HISC 107
IF(IAROS(4)-FIX(IAROS(2)/ARCS(3)+.5).GT.NCOL) GO TO 60	HISC 108

CALL ADDRESS(4,K1)	MISC 109
IF(K1.LE.0) GO TO 60	MISC 110
K1=K1-1	MISC 111
IF(KIND(1).NE.0) GO TO 460	MISC 112
CALL ADDRESS(1,IAROS(1))	MISC 113
IF(IAROS(1).LE.0) GO TO 60	MISC 114
N = IAROS(1) - 1	MISC 115
DO 450 I=1,NRMAX	MISC 116
J = K + I	MISC 117
450 A(I) = RC(J)	MISC 118
GO TO 480	MISC 119
460 DO 470 I=1,NRMAX	MISC 120
470 A(I) = AROS(I)	MISC 121
480 DO 500 I=2,3	MISC 122
IF(KIND(I).EQ.0) AROS(I)=IAROS(I)	MISC 123
500 CONTINUE	MISC 124
IF(IAROS(2)*AROS(3).LE.0.) GO TO 60	MISC 125
IF(NRMAX.LT.1) GO TO 120	MISC 126
CALL PLBK	MISC 127
IF(NFLAG.EQ.1) RETURN	MISC 128
CC = AROS(3)	MISC 129
570 DO 580 I=1,NRMAX	MISC 130
K = K1 + I	MISC 131
580 RC(K) = FEXP2(A(I),CC)	MISC 132
IF(ABS(CC)-ABS(IAROS(2)).GE.0.) GO TO 30	MISC 133
CC = CC + AROS(3)	MISC 134
IAROS(4) = IAROS(4) + 1	MISC 135
CALL ADDRESS(4,K1)	MISC 136
IF(K1.LE.0) GO TO 30	MISC 137
K1=K1-1	MISC 138
GO TO 570	MISC 139
C	MISC 140
C	MISC 141
C	MISC 142
600 IF(IAROS.NE.7) GO TO 10	MISC 143
CALL CHKIND(J)	MISC 144
IF(J.NE.0) GO TO 60	MISC 145
IAROS(9)=IAROS(1)	MISC 146
DO 630 I=2,7	MISC 147
630 IAROS(I-1)=IAROS(I)	MISC 148
IAROS(7)=IAROS(8)+IAROS(3)	MISC 149
IAROS(8)=IAROS(4)	MISC 150
J=2	MISC 151
CALL HTXCHK(J)	MISC 152
IF(J-1) 660,60,640	MISC 153
640 K = 17	MISC 154
GO TO 20	MISC 155
660 IF(IAROS(9).LT.1) GO TO 60	MISC 156
CALL PLBK	MISC 157
IF(NFLAG.EQ.1) RETURN	MISC 158
IY=IAROS(1)	MISC 159
IENO = IAROS(9)	MISC 160
LONG = IAROS(3)	MISC 161
LWIDE = IAROS(4)	MISC 162

```

      DO 705 I=1,LNIDE
      K1=IY
      DO 700 K=1,LONG
      A(K)=RC(K1)
700 K1=K1+1
      CALL SCRAM(I,2)
705 IY = IY + NROW
      IY=IARG6(5)
      DO 730 JJ = 1, IEND
      IX=IY
      DO 720 J=1,LNIDE
      CALL SCRAM(I,J)
      K1=IX
      DO 710 K=1,LONG
      RC(K1)=A(K)
710 K1=K1+1
720 IX = IX + NROW
730 IY = IY + LONG
      GO TO 30
      END

```

```

HISC 163
HISC 164
HISC 165
HISC 166
HISC 167
HISC 168
HISC 169
HISC 170
HISC 171
HISC 172
HISC 173
HISC 174
HISC 175
HISC 176
HISC 177
HISC 178
HISC 179
HISC 180
HISC 181
HISC 182

```

SUBROUTINE MMULT		MMUL	1
COMMON / BLOCKA/MODE,M,KARD(77),KARC,ARG,ARG2,NEWCO(19),KROEND		MMUL	2
1,NEWCO(19,51),KSAVE,NSAVE,NFLAG		MMUL	3
COMMON / BLOCKD / RC(2433),IARCS(69),KIND(33),AROTAB(51),NRMAX,		MMUL	4
1 NR0W,NCOL,NARCS,VWXYZ(5)		MMUL	5
COMMON/SCRAT/RI(80)		MMUL	6
C	*****	MMUL	7
C	THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMAND MMULT.	MMUL	8
C	*****	MMUL	9
	IR0WA=IARCS(9)	MMUL	10
C	*****	MMUL	11
C	CHECK TO SEE IF WE HAVE CORRECT NUMBER OF ARGUMENTS	MMUL	12
C	*****	MMUL	13
	IF(IARCS.GT.10.OR.NARCS.LT.7) GO TO 0110	MMUL	14
C	*****	MMUL	15
C	CHECK TO SEE IF ALL ARGUMENTS ARE INTEGERS	MMUL	16
C	*****	MMUL	17
	600 J=NARCS	MMUL	18
	CALL CKIND(J)	MMUL	19
	IF(J.NE.0) GO TO 0103	MMUL	20
C	*****	MMUL	21
C	CHECK TO SEE IF DIMENSIONS ARE CORRECT	MMUL	22
C	*****	MMUL	23
	IF(NARCS.EQ.10) GO TO 040	MMUL	24
	IARCS(10)=IARCS(NARCS)	MMUL	25
	IARCS(9)=IARCS(NARCS-1)	MMUL	26
	IF(NARCS.EQ.9) GO TO 030	MMUL	27
	IARCS(9)=IR0WA	MMUL	28
	IF(NARCS.EQ.7) GO TO 020	MMUL	29
	IF(IARCS(6).NE.IR0WA) GO TO 0103	MMUL	30
	020 IARCS(6)=IARCS(5)	MMUL	31
	IARCS(5)=IARCS(4)	MMUL	32
	IARCS(4)=IR0WA	MMUL	33
	GO TO 030	MMUL	34
	030 IARCS(8)=IARCS(7)	MMUL	35
	031 IARCS(7)=IARCS(4)	MMUL	36
	GO TO 1100	MMUL	37
	040 IF(IARCS(4).NE.IARCS(7)) GO TO 0103	MMUL	38
	1100 ICOLA=IARCS(9)	MMUL	39
	IF(IR0WA.GT.15.OR.ICOLA.GT.15) GO TO 0124	MMUL	40
	IARCS(12)=ICOLA	MMUL	41
	IARCS(11)=IR0WA	MMUL	42
	J=9	MMUL	43
	CALL HTXCHK(J)	MMUL	44
	IF(J-1) 1200,0103,0117	MMUL	45
	1200 CALL PLCK	MMUL	46
	IF(NFLAG.EQ.1) RETURN	MMUL	47
C	*****	MMUL	48
C	BEGIN MULTIPLICATION	MMUL	49
C	*****	MMUL	50
	ICOLA=IARCS(4)	MMUL	51
	IUP=IARCS(5)	MMUL	52
	GO 3040 ICB=1,ICOLA	MMUL	53
	IAP=IARCS(1)	MMUL	54

```

DO 9020 IRA=1,IRONA
IA=IAP
IB=IBP
A(IRA)=0.
DO 9000 J=1,ICOLA
A(IRA)=RC(IA)+RC(1B)+A(IRA)
IA=IA+IRON
IB=IB+1
9000 CONTINUE
9020 IAP=IAP+1
CALL SCRAM(10,2)
9040 ICP=ICP+IRON
C *****
C STORE MATRIX PRODUCT
C *****
ICP=IARCS(9)
DO 0100 J=1,ICOLA
IC=ICP
CALL SCRAM(J,1)
DO 0020 I=1,IRONA
RC(IC)=A(I)
IC=IC+1
0080 CONTINUE
0100 ICP=ICP+IRON
RETURN
0109 CALL ERROR(9)
RETURN
0110 CALL ERROR(10)
RETURN
0117 CALL ERROR(17)
RETURN
0124 CALL ERROR(24)
RETURN
END

```

```

MHUL 55
MHUL 56
MHUL 57
MHUL 58
MHUL 59
MHUL 60
MHUL 61
MHUL 62
MHUL 63
MHUL 64
MHUL 65
MHUL 66
MHUL 67
MHUL 68
MHUL 69
MHUL 70
MHUL 71
MHUL 72
MHUL 73
MHUL 74
MHUL 75
MHUL 76
MHUL 77
MHUL 78
MHUL 79
MHUL 80
MHUL 81
MHUL 82
MHUL 83
MHUL 84
MHUL 85
MHUL 86
MHUL 87
MHUL 88

```

C 37 96 SUBROUTINE MOP

7 10 73

SUBROUTINE MOP

COMMON / BLOCKA/MODE,M,KARD(77),KARG,ARG,ARG2,NEHCD(19),KRDEND
1,NEHCOS(19,10),KSAVE,NSAVE,NFLAG

COMMON / BLOCKD / AC(2439),IARGS(69),KIND(39),ARCTAB(51),NMAX,

1 NROW,NCOL,NARGS,VWXYZ(5)

DIMENSION ARGS(1)

EQUIVALENCE (NARGS(1),AC(2401))

COMMON/BLOCKE/NAME(4),L1,L2,ISAF LG

COMMON/SCAT/A(2400),NS

DATA ONE/1.0/,ZERO/0.0/

C****

C**** THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS HDEFINE,
C**** ADEFINE, NZERO, AZERO, MERASE, AERASE, MIDENT, MDIAG, ADIAG AND MT

C**** L2=1 HDEFINE,ADEFINE

C**** L2=2 MDIAG,ADIAG

C**** L2=3 NZERO,AZERO,MERASE,AERASE

C**** L2=4 MIDENT

C**** L2=5 MTRACE

C****

GO TO (100,100,180,180,150,150,150,150,160,250),L2

100 IF (NARGS.NE.4.AND.NARGS.NE.5) GO TO 10

IF (KIND(NARGS).NE.1) GO TO 3

IF (NARGS.EQ.4) IARGS(4)=IARGS(3)

CONST=ARGS(NARGS)

CONST=ARGS(NARGS)

J=NARGS-1

105 CALL CKIND (J)

IF (J.NE.0) GO TO 3

J=1

IF (L2.EQ.10) J=2

CALL HTXCHK (J)

IF (J.NE.0) GO TO 17

CALL PLBK

IF (NFLAG.EQ.1) RETURN

JB=IARGS(1)

IF (L2.EQ.10) GO TO 260

N=IARGS(3)

K=IARGS(4)

JA=JB

DO 120 KA=1,K

JC=JB

DO 110 NA=1,N

AC(JC)=CONST

110 JC=JC+1

IF (KA.GT.N) GO TO 120

AC(JA)=CONST

JA=JA+NROW+1

120 JB=JB+NROW

IF (L2.EQ.4.OR.L2.EQ.9) GO TO 180

RETURN

150 IF (NARGS.NE.3.AND.NARGS.NE.4) GO TO 10

CONST=ZERO

CONST=ZERO

```

J=NARGS
IF (NARGS.EQ.4) GO TO 105
IARGS(4)=IARGS(3)
GO TO 105
160 CONST=ZERO
CONST=ONE
J=NARGS
IF (NARGS.NE.3) GO TO 170
IARGS(4)=IARGS(3)
GO TO 105
170 IF (IARGS(3).NE.IARGS(4)) GO TO 3
GO TO 105
180 J=NARGS-1
CONST=ZERO
CONST=ZERO
IF (NARGS.NE.4.AND.NARGS.NE.5) GO TO 10
CALL ADDRESS (NARGS,N)
IF (N) 180,11,184
184 N=IARGS(3)
DO 183 NA=1,N
A(NA)=AC(N)
186 N=N+1
188 IF (NARGS.EQ.5) GO TO 170
IARGS(5)=IARGS(4)
IARGS(4)=IARGS(3)
GO TO 105
190 JB=IARGS(1)
IF (KIND(NARGS).EQ.0) GO TO 220
CONST=NARGS(NARGS)
DO 200 NA=1,N
AC(JB)=CONST
200 JB=JB+1+NROW
RETURN
220 DO 230 NA=1,N
AC(JB)=A(NA)
230 JB=JB+1+NROW
RETURN
250 IARGS(7)=1
IARGS(8)=1
J=NARGS
IF (NARGS.NE.6.AND.NARGS.NE.5) GO TO 10
IF (NARGS.EQ.6) GO TO 105
IARGS(6)=IARGS(5)
IARGS(5)=IARGS(4)
IARGS(4)=IARGS(3)
GO TO 105
260 TRACE=0.
N=KIND(IARGS(3),IARGS(4))
DO 270 NA=1,N
TRACE=TRACE+AC(JB)
270 JB=JB+NROW+1
ICX=IARGS(5)
AC(ICX)=TRACE
RETURN

```

```
3 CALL ERROR(3)
  RETURN
10 CALL ERROR(10)
  RETURN
17 CALL ERROR(17)
  RETURN
11 CALL ERROR(11)
  RETURN
END
```

```

SUBROUTINE MOVE
COMMON / BLOCKA/MODE,H,KORD(77),KARG,ARG,ARG2,NEHCO(19),KROEND
1,NEHCO(19,5),KRAVE,MSHVE,NFLAG
COMMON / BLOCKD / RC(2499),IARG(69),KIND(39),ARGTAG(51),NRMAX,
1 NR0W,NCOL,NARG,VMXYZ(15)
COMMON/SCRAT/RI(80)
DIMENSION ARG(11)
EQUIVALENCE(ARG(1),RC(2401)),(19,IARG(9)),(14,IARG(4))

C
C
C
C
THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS MOVE,
BLOCKTRANSFER, AMOVE AND HMOVE.

IF( NARGS .EQ. 6 ) GO TO 70
K = 10
20 CALL ERROR( K )
20 RETURN
40 K = 20
GO TO 10
50 K = 17
GO TO 10
60 K=3
GO TO 10
70 J=3
CALL CRIND(J)
IF(J.NE.0) GO TO 40
IARG(7)=19
IARG(8)=14
J=2
CALL HTXCHK(J)
IF(J=1) 80,90,50
80 CALL PLBK
IF(NFLAG.EQ.1) RETURN
K=IARG(1)
DO 110 I=1,14
KK=K
DO 100 II=1,19
A(II)=RC(KK)
100 KK=KK+1
CALL SCRAN(1,2)
110 K=K+NR0W
K=IARG(5)
DO 210 I=1,14
KK=K
CALL SCRAN(1,1)
DO 200 II=1,19
RC(KK)=A(II)
200 KK=KK+1
210 K=K+NR0W
GO TO 20
END

```

```

MOVE 1
MOVE 2
MOVE 3
MOVE 4
MOVE 5
MOVE 6
MOVE 7
MOVE 8
MOVE 9
MOVE 10
MOVE 11
MOVE 12
MOVE 13
MOVE 14
MOVE 15
MOVE 16
MOVE 17
MOVE 18
MOVE 19
MOVE 20
MOVE 21
MOVE 22
MOVE 23
MOVE 24
MOVE 25
MOVE 26
MOVE 27
MOVE 28
MOVE 29
MOVE 30
MOVE 31
MOVE 32
MOVE 33
MOVE 34
MOVE 35
MOVE 36
MOVE 37
MOVE 38
MOVE 39
MOVE 40
MOVE 41
MOVE 42
MOVE 43
MOVE 44
MOVE 45
MOVE 46
MOVE 47
MOVE 48
MOVE 49
MOVE 50

```

SUBROUTINE MRAIG	MRAI	1
COMMON / BLOCKA/MSOE,M,KARD(77),KARG,ARG,ARG2,NENCO(19),KROEND	MRAI	2
1,NENCO6(19,5),KSAVE,NSAVE,NFLAG	MRAI	3
COMMON / BLOCKB / RC(2439),IARG6(69),KIND(39),AROTAB(51),NRMAX,	MRAI	4
1 NRON,NCOL,NARG6,VHXYE(5)	MRAI	5
DIMENSION ARG6(1)	MRAI	6
EQUIVALENCE(AROS(1),RC(2401))	MRAI	7
COMMON/SCRAT/AL(50)	MRAI	8
C =====	MRAI	9
C THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMAND MRAIG.	MRAI	10
C =====	MRAI	11
ISIZE=IARG6(3)	MRAI	12
C =====	MRAI	13
C CHECK NUMBER OF ARGUMENTS	MRAI	14
C =====	MRAI	15
IF(NARG6.NE.7.AND.NARG6.NE.6) GO TO 10	MRAI	16
C =====	MRAI	17
C CHECK TO SEE IF ALL ARGUMENTS ARE INTEGER	MRAI	18
C =====	MRAI	19
J=NARG6-2	MRAI	20
IF(KIND(J).NE.0) GO TO 620	MRAI	21
200 IARG6(9)=IARG6(J)	MRAI	22
IF(IARG6(9).LT.1) GO TO 9	MRAI	23
J=NARG6	MRAI	24
CALL CKIND(J)	MRAI	25
IF(J.EQ.0) GO TO 800	MRAI	26
9 CALL ERROR (3)	MRAI	27
RETURN	MRAI	28
620 IARG6(J)=AROS(J)	MRAI	29
KIND(J)=0	MRAI	30
GO TO 200	MRAI	31
C =====	MRAI	32
C CHECK TO SEE IF DIMENSIONS ARE CORRECT	MRAI	33
C =====	MRAI	34
900 IF(NARG6.EQ.0) GO TO 1100	MRAI	35
IF(IARG6(9).NE.IARG6(4)) GO TO 9	MRAI	36
IARG6(5)=IARG6(6)	MRAI	37
IARG6(6)=IARG6(7)	MRAI	38
GO TO 1100	MRAI	39
1100 IARG6(4)=ISIZE	MRAI	40
1150 IARG6(7)=ISIZE	MRAI	41
IARG6(8)=ISIZE	MRAI	42
J=2	MRAI	43
CALL MTXCHK(J)	MRAI	44
IF(J-1) 1200,9,17	MRAI	45
1200 CALL FLBK	MRAI	46
IF(NFLAG.EQ.1) RETURN	MRAI	47
C =====	MRAI	48
C BEGIN MULTIPLICATION	MRAI	49
C =====	MRAI	50
NPOW=IARG6(5)-1	MRAI	51
IF(NPOW.GE.1) GO TO 4050	MRAI	52
IF(IARG6(1).EQ.IARG6(6)) RETURN	MRAI	53
IRP=IARG6(1)	MRAI	54

ISAVP=IAROS(5)	NRAI 55
DO 4040 I=1,ISIZE	NRAI 56
ISAV=ISAVP	NRAI 57
IP=IRP	NRAI 58
DO 4030 J=1,ISIZE	NRAI 59
RC(ISAV)=RC(IP)	NRAI 60
IP=IP+NRON	NRAI 61
4030 ISAV=ISAV+NRON	NRAI 62
ISAVP=ISAVP+1	NRAI 63
4040 IRP=IRP+1	NRAI 64
RETURN	NRAI 65
4050 DO 5040 K=1,NPCN	NRAI 66
ISAVP=IAROS(5)	NRAI 67
IF(K.GT.1) GO TO 4060	NRAI 68
IRP=IAROS(1)	NRAI 69
GO TO 4070	NRAI 70
4060 IRP=IAROS(5)	NRAI 71
4070 DO 5040 I=1,ISIZE	NRAI 72
IP=IAROS(1)	NRAI 73
ISAV=ISAVP	NRAI 74
IZ=IRP	NRAI 75
C *****	NRAI 76
C GAVE ROW OF MATRIX	NRAI 77
C *****	NRAI 78
DO 4000 J=1,ISIZE	NRAI 79
A(J)=RC(IZ)	NRAI 80
IZ=IZ+NRON	NRAI 81
4000 CONTINUE	NRAI 82
DO 5020 J=1,ISIZE	NRAI 83
IC=IP	NRAI 84
RC(ISAV)=0.	NRAI 85
DO 5000 JP=1,ISIZE	NRAI 86
RC(ISAV)=RC(ISAV)+A(JP)*RC(IC)	NRAI 87
IC=IC+1	NRAI 88
5000 CONTINUE	NRAI 89
ISAV=ISAV+NRON	NRAI 90
IP=IP+NRON	NRAI 91
5020 CONTINUE	NRAI 92
ISAVP=ISAVP+1	NRAI 93
IRP=IRP+1	NRAI 94
5040 CONTINUE	NRAI 95
RETURN	NRAI 96
10 CALL ERROR(10)	NRAI 97
RETURN	NRAI 98
17 CALL ERROR(17)	NRAI 99
RETURN	NRAI 100
END	NRAI 101

	SUBROUTINE HSCROW	HSCR	1
	COMMON / BLOCKA/MODE,H,NAR0(77),NAR0,AR0,AR02,HENCO(10),KROEND	HSCR	2
	1,HENCO6(10,5),KSAVE,NSAVE,NFLAO	HSCR	3
	COMMON / BLOCKD / RC(2459),IAR06(99),KIND(39),AROTAD(51),NRMAX,	HSCR	4
	1,NROW,NCOL,NAR06,VWXYZ(5)	HSCR	5
	COMMON/BLOCKE/NAME(4),L1,L2,ISRFLO	HSCR	6
	EQUIVALENCE (12,IAR06(2)),(13,IAR06(9))	HSCR	7
C		HSCR	8
C	THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS PARSUM,	HSCR	9
C	PARPROD, RMS, AVERAGE AND SUM.	HSCR	10
C		HSCR	11
	ELEN = 0.0	HSCR	12
	IF(NAR06.EQ.2) GO TO 40	HSCR	13
10	CALL ERROR(10)	HSCR	14
	RETURN	HSCR	15
40	CALL ADDRESS(1, J1)	HSCR	16
	IF(J1.GT.0) GO TO 60	HSCR	17
50	CALL ERROR(3)	HSCR	18
	RETURN	HSCR	19
60	CALL ADDRESS(NAR06, J2)	HSCR	20
	IF(J2.LE.0) GO TO 50	HSCR	21
	IF(NRMAX.GT.0) GO TO 140	HSCR	22
	CALL ERROR(0)	HSCR	23
	RETURN	HSCR	24
140	IF(NAR06.LT.3) GO TO 200	HSCR	25
	IF(L2.NE.5) GO TO 10	HSCR	26
	NAR01=NAR06-1	HSCR	27
	DO 100 I=2,NAR01	HSCR	28
	IF (KIND(1) .NE. 0) GO TO 50	HSCR	29
	IF(IAR06(1).LE.0.OR.IAR06(1).GT.NROW) GO TO 50	HSCR	30
100	CONTINUE	HSCR	31
	IF(12.GT.13.AND.NAR06.EQ.4) GO TO 50	HSCR	32
	CALL PLOK	HSCR	33
	IF(NFLAO.EQ.1) RETURN	HSCR	34
	IF(NAR06.GT.4) GO TO 170	HSCR	35
C		HSCR	36
C	SUM FROM ROW .. TO ROW ..	HSCR	37
C		HSCR	38
	J=J1-1	HSCR	39
	DO 150 JJ=12,13	HSCR	40
	JJJ=J+JJ	HSCR	41
150	ELEN=ELEN+RC(JJJ)	HSCR	42
160	CALL VECTOR (ELEN,J2)	HSCR	43
	RETURN	HSCR	44
C		HSCR	45
C	SUM DISCRETE ROWS	HSCR	46
C		HSCR	47
170	DO 190 I=2,NAR01	HSCR	48
	J = J1 + IAR06(1)	HSCR	49
190	ELEN = ELEN + RC(J - 1)	HSCR	50
	GO TO 160	HSCR	51
200	CALL PLOK	HSCR	52
	IF(NFLAO.EQ.1) RETURN	HSCR	53
	IF(NRMAX.LE.0) GO TO 140	HSCR	54

	FNHMAX=NRHMAX	MSCR	85
C		MSCR	86
C	PARSUM, PARPRODUCT	MSCR	87
C		MSCR	88
	IF(L2 - 3) 220, 280, 300	MSCR	89
220	J = L2 - 1	MSCR	90
	RC(J2) = RC(J1)	MSCR	91
	IF(NRHMAX.EQ.1) RETURN	MSCR	92
	GO 240 I = 2, NRHMAX	MSCR	93
	J1 = J1 + 1	MSCR	94
	J2 = J2 + 1	MSCR	95
	IF(J.EQ.0) GO TO 230	MSCR	96
	RC(J2) = RC(J2 - 1) + RC(J1)	MSCR	97
	GO TO 240	MSCR	98
230	RC(J2) = RC(J2 - 1) + RC(J1)	MSCR	99
240	CONTINUE	MSCR	100
	RETURN	MSCR	101
C		MSCR	102
C	RMS	MSCR	103
C		MSCR	104
280	GO 290 I = 1, NRHMAX	MSCR	105
	J = J1 + 1	MSCR	106
290	ELEN = ELEN + RC(J - 1) * 2	MSCR	107
	ELEN = F60RT(ELEN/FNHMAX)	MSCR	108
	GO TO 180	MSCR	109
C		MSCR	110
C	AVERAGE, SUM ENTIRE ROW	MSCR	111
C		MSCR	112
300	GO 310 I = 1, NRHMAX	MSCR	113
	J = J1 + 1	MSCR	114
310	ELEN = ELEN + RC(J - 1)	MSCR	115
	IF(L2.EQ.6) GO TO 180	MSCR	116
	ELEN=ELEN/FNHMAX	MSCR	117
	GO TO 180	MSCR	118
	END	MSCR	119

	SUBROUTINE MTXCHK(J)	MTXC	1
	COMMON / BLOCK / RC(2499),IAROS(69),KIND(99),ARCTAB(51),NRMAX,	MTXC	2
	1 NROW,NCOL,NAROS,VNXYE(5)	MTXC	3
C		MTXC	4
C	THIS SUBROUTINE IS USED TO CHECK TO SEE IF SPECIFIED MATRICES	MTXC	5
C	ARE LEGAL.	MTXC	6
C	J AS INPUT = NO OF MATRICES TO BE CHECKED	MTXC	7
C	IAROS(1), IAROS(5),....,IAROS(4*(J-1)+1) STARTING ROW OF MAT	MTXC	8
C	IAROS(2), IAROS(6),....,IAROS(4*(J-1)+2) STARTING COLUMN OF MAT	MTXC	9
C	IAROS(3), IAROS(7),....,IAROS(4*(J-1)+3) NO. OF ROWS	MTXC	10
C	IAROS(4), IAROS(8),....,IAROS(4*(J-1)+4) NO OF COLUMNS	MTXC	11
C	UPON RETURN	MTXC	12
C	J=0 IF ALL MATRICES ARE IN WORK SHEET	MTXC	13
C	AND	MTXC	14
C	IAROS(1),IAROS(5),....,IAROS(4*(J-1)+1) WILL CONTAIN STARTING	MTXC	15
C	ADDRESS OF MATRIX	MTXC	16
C	J=0 ZERO IF MATRIX IS NOT IN WORK SHEET	MTXC	17
C	J=1 SOME IAROS ARE NEGATIVE. J=2 MATRIX TOO BIG FOR WORKSHEET	MTXC	18
C		MTXC	19
	JB=4*J	MTXC	20
	J = 0	MTXC	21
	DO 100 I=1,JB	MTXC	22
	IF(IAROS(I).GT.0) GO TO 100	MTXC	23
	J=1	MTXC	24
	RETURN	MTXC	25
100	CONTINUE	MTXC	26
	DO 120 I=1,JB,4	MTXC	27
	IF(IAROS(I)+IAROS(I+2)-1.GT.NROW) GO TO 130	MTXC	28
	IF(IAROS(I+1)+IAROS(I+3)-1.GT.NCOL) GO TO 130	MTXC	29
	KIND(I+1)=0	MTXC	30
	CALL ADRESS(I+1,JC)	MTXC	31
120	IAROS(I)=JC+IAROS(I)-1	MTXC	32
	RETURN	MTXC	33
130	J=2	MTXC	34
	RETURN	MTXC	35
	END	MTXC	36

SUBROUTINE MXTX	MXTX	1
COMMON / BLOCKA/MODE,H,KARO(77),KARO,ARG,ARG2,NENCO(10),KRVEND	MXTX	2
1,NENCO\$(10,51),KSAVE,NSAVE,NFLAD	MXTX	3
COMMON / BLOCKD / RC(2499),IARG6(69),KIND(99),ARGTAB(51),NRHMX,	MXTX	4
1 NR0H,NCOL,NARCS,VHXYZ(5)	MXTX	5
COMMON/BLOCKE/NAME(4),L1,L2,ISRFLO	MXTX	6
COMMON / BLOCKF / NCTOP	MXTX	7
COMMON/SCRAT/AC(50)	MXTX	8
C *****	MXTX	9
C THIS SUBROUTINE IS USED TO EXECUTE THE COMMANDS M(X'') AND M(X'X).	MXTX	10
C L2 = 1 - M(X'')	MXTX	11
C L2 = 2 - M(X'X)	MXTX	12
C *****	MXTX	13
C CHECK FOR CORRECT NUMBER OF ARGUMENTS	MXTX	14
C DECIDE WHETHER COMMAND IS M(X'X'') OR M(X'X')	MXTX	15
C L2 = 3 MEANS M(X'X'') L2 = 2, NARCS .GT. 6 MEANS M(X'X')	MXTX	16
C *****	MXTX	17
GO TO (100,10,20,40,40,60,60),L2	MXTX	18
10 IF(NARCS .LE. 6) GO TO 100	MXTX	19
20 L2 = 4 - L2	MXTX	20
CALL TRANSF	MXTX	21
RETURN	MXTX	22
40 CALL NOAHAD	MXTX	23
RETURN	MXTX	24
60 CALL ARYVEC	MXTX	25
RETURN	MXTX	26
100 IF(NARCS .NE. 5 .AND. NARCS .NE. 6) GO TO 210	MXTX	27
C *****	MXTX	28
C CHECK TO SEE IF ALL ARGUMENTS ARE INTEGERS	MXTX	29
C *****	MXTX	30
J=NARCS	MXTX	31
CALL CHKIN(J)	MXTX	32
IF(J.NE.0) GO TO 220	MXTX	33
C *****	MXTX	34
C CHECK TO SEE IF DIMENSIONS ARE OUT OF RANGE	MXTX	35
C COMPUTE ADDRESSES	MXTX	36
C *****	MXTX	37
IF(NARCS.EQ. 6) GO TO 120	MXTX	38
IARCS(0) = IARCS(0)	MXTX	39
IARCS(1) = IARCS(1)	MXTX	40
IARCS(4) = IARCS(4)	MXTX	41
120 IARCS(0)=IARCS(L2+2)	MXTX	42
IARCS(7)=IARCS(0)	MXTX	43
IF(IARCS(0).LE.10) GO TO 200	MXTX	44
CALL ERROR(24)	MXTX	45
RETURN	MXTX	46
200 J=2	MXTX	47
CALL MTCCHK(J)	MXTX	48
IF(J-1) 200, 220, 240	MXTX	49
210 CALL ERROR(10)	MXTX	50
RETURN	MXTX	51
220 CALL ERROR(5)	MXTX	52
RETURN	MXTX	53
240 CALL ERROR(17)	MXTX	54

```

      RETURN
260 CALL PLOK
   IF(NFLAQ.EQ.1) RETURN
   IF(L2.EN.2) GO TO 320
   IP=IAROS(3)
   JP=IAROS(4)
   IADD1=NRON
   IADD2=1
   GO TO 340
320 IP=IAROS(4)
   JP=IAROS(3)
   IADD1=1
   IADD2=NRON
340 IOP=IAROS(1)
   DO 440 K=1,IP
   IAP=IAROS(1)
   IC=1
   DO 420 I=1,IP
   A(IC)=Q,
   IA=IAP
   IB=IBP
   DO 400 J=1,JP
   A(IC)=RC(IA)+RC(IB)+A(IC)
   IA=IA+IADD1
   IB=IB+IADD1
400 CONTINUE
   IAP=IAP+IADD2
   IC=IC+1
420 CONTINUE
   IBP=IBP+IADD2
440 CALL SCRAM(K,2)
C =====
Q MOVE FROM SCRATCH AREA TO STORAGE
C =====
   IC = IAROS( 5 )
   DO 520 I=1,IP
   IB=1
   CALL SCRAM(I,1)
   DO 500 J=1,JP
   RC(IC)=A(IB)
   IB=IB+1
   IC=IC+1
500 CONTINUE
   IC = IC + ( NRON+NCTOP-1 ) - IP
520 CONTINUE
      RETURN
      END

```

```

NXTX 56
NXTX 56
NXTX 57
NXTX 58
NXTX 59
NXTX 60
NXTX 61
NXTX 62
NXTX 63
NXTX 64
NXTX 65
NXTX 66
NXTX 67
NXTX 68
NXTX 69
NXTX 70
NXTX 71
NXTX 72
NXTX 73
NXTX 74
NXTX 75
NXTX 76
NXTX 77
NXTX 78
NXTX 79
NXTX 80
NXTX 81
NXTX 82
NXTX 83
NXTX 84
NXTX 85
NXTX 86
NXTX 87
NXTX 88
NXTX 89
NXTX 90
NXTX 91
NXTX 92
NXTX 93
NXTX 94
NXTX 95
NXTX 96
NXTX 97
NXTX 98
NXTX 99
NXTX 100
NXTX 101

```

SUBROUTINE NNAME(NAME)
 COMMON / BLOCKA/MODE,M,KARO(77),KARO,ARO,ARO2,MENCO(19),KROEND
 1,MENCO6(10,6),KSAVE,NSAVE,NFLAG
 DIMENSION NAME(2),MISC(6)

NNAM 1
 NNAM 2
 NNAM 3
 NNAM 4
 NNAM 5
 NNAM 6
 NNAM 7
 NNAM 8
 NNAM 9
 NNAM 10
 NNAM 11
 NNAM 12
 NNAM 13
 NNAM 14
 NNAM 15
 NNAM 16
 NNAM 17
 NNAM 18
 NNAM 19
 NNAM 20
 NNAM 21
 NNAM 22
 NNAM 23
 NNAM 24
 NNAM 25
 NNAM 26
 NNAM 27
 NNAM 28
 NNAM 29
 NNAM 30
 NNAM 31
 NNAM 32
 NNAM 33
 NNAM 34
 NNAM 35
 NNAM 36
 NNAM 37
 NNAM 38
 NNAM 39
 NNAM 40
 NNAM 41
 NNAM 42
 NNAM 43
 NNAM 44
 NNAM 45
 NNAM 46
 NNAM 47
 NNAM 48
 NNAM 49
 NNAM 50
 NNAM 51
 NNAM 52
 NNAM 53
 NNAM 54

C THIS SUBROUTINE ASSEMBLES A NAME UP TO THE FIRST NON-LETTER OR UP TO
 C SIX LETTER, WHICHEVER IS FIRST. THE INDEX, M, IS INITIALLY POINTING AT
 C THE FIRST LETTER. IT IS LEFT POINTING AT THE FIRST NON-LETTER.
 C THE FIRST THREE CHARACTERS GO INTO THE FIRST WORD OF NAME
 C THE SECOND THREE CHARACTERS GO INTO THE SECOND WORD OF NAME
 C CONTINUE

CONVERSION TABLE FOR ALPHABETIC TO NUMERIC AS USED BY OMNITAD.

A	729	27	1
B	1450	54	2
C	2107	81	3
D	2916	108	4
E	3645	135	5
F	4374	162	6
G	5103	189	7
H	5832	216	8
I	6561	243	9
J	7290	270	10
K	8019	297	11
L	8748	324	12
M	9477	351	13
N	10206	378	14
O	10935	405	15
P	11664	432	16
Q	12393	459	17
R	13122	486	18
S	13851	513	19
T	14580	540	20
U	15309	567	21
V	16038	594	22
W	16767	621	23
X	17496	648	24
Y	18225	675	25
Z	18954	702	26

DO 10 I=1,6
 MISC(I)=0
 DO 20 I=1,6
 L=KARO(M)-3
 IF(L.LT.1.OR.L.GE.27)GO TO 40
 MISC(I)=L
 M=M+1
 IF(KARO(M).LT.10.OR.KARO(M).GE.36)GO TO 40
 M=M+1
 GO TO 30
 40 NAME(1)=MISC(3)+27*(MISC(2)+27*MISC(1))
 NAME(2)=MISC(6)+27*(MISC(5)+27*MISC(4))
 RETURN
 END

	FUNCTION NONBLA(I)	NONB	1
	COMMON / BLOCKA/MODE,H,KARD(77),KARG,ARG,ARG2,NEHCO(19),KROEND	NONB	2
	1,NEHCDS(19,5),KSAVE,NSAVE,NFLAO	NONB	3
C	THIS FUNCTION SUBPROGRAM SCANS THE ARRAY KARD STARTING WITH THE	NONB	4
C	NTH ELEMENT UNTIL A NON-BLANK CHARACTER IS FOUND. N WILL POINT	NONB	5
C	AT IT AND THE VALUE OF N WILL BE RETURNED AS THE FUNCTIONAL VALUE.	NONB	6
		NONB	7
	1 IF(KARD(N).NE.44100 TO 2	NONB	8
	N=N+1	NONB	9
	GO TO 1	NONB	10
	2 NONBLA=KARD(N)	NONB	11
	RETURN	NONB	12
	END	NONB	13

ONCONV	START		ONCO	1
	USING =,15		ONCO	2
	SAVE (14,12)		ONCO	3
	L 3.0(.1) ADDRESS OF NHCO IN 3		ONCO	4
	L 4.4(.1) ADDRESS OF KRD IN 4		ONCO	5
	HVC 0(80.4).0(3)		ONCO	6
	VR 0(80.4).TABLE		ONCO	7
	LA 5.0		ONCO	8
	LA 0.4		ONCO	9
	LA 7.1		ONCO	10
	L 9.8(.1) ADDRESS OF KRDEND IN 8		ONCO	11
	L 11.0(.9) KRDEND IN 11		ONCO	12
	LR 6.11		ONCO	13
	SR 6.7 KRDEND - 1 IN 8		ONCO	14
	NR 10.8 4=KRDEND IN 11		ONCO	15
	SR 11.8 (4-1)=KRDEND IN 11		ONCO	16
LOOP	IC 5.0(0.4)		ONCO	17
	ST 5.0(11.4)		ONCO	18
	SR 11.8		ONCO	19
	BCT 6,LOOP		ONCO	20
	IC 5.0(6.4)		ONCO	21
	ST 5.0(11.4)		ONCO	22
	RETURN (14,12)		ONCO	23
TABLE	DC 256X'2E'		ONCO	24
	ORG TABLE+C'A'		ONCO	25
A	DC 8AL1(M-A+10)		ONCO	26
	ORG TABLE+C'J'		ONCO	27
J	DC 8AL1(M-J+10)		ONCO	28
	ORG TABLE+C'S'		ONCO	29
S	DC 8AL1(M-S+20)		ONCO	30
	ORG TABLE+C'O'		ONCO	31
ZERO	DC 10AL1(M-ZERO)		ONCO	32
	ORG TABLE+C'.'		ONCO	33
	DC AL1(44)		ONCO	34
	ORG TABLE+C'.'		ONCO	35
	DC AL1(37.40.41.39)		ONCO	36
	ORG TABLE+C'N'		ONCO	37
	DC AL1(40.42.46.40.39.38)		ONCO	38
	ORG TABLE+C'.'		ONCO	39
	DC AL1(43)		ONCO	40
	ORG TABLE+125		ONCO	41
	DC AL1(40.45)		ONCO	42
	END		ONCO	43

SUBROUTINE OMNIT	OMNI 1
EXTERNAL PFINT, EODINT	OMNI 2
COMMON / KPLT / NFRAM, KKND, SIZE, SPACE	OMNI 3
COMMON / BLOCKA / MODE, H, KARO(77), KARO, ARG, ARG2, NEWCO(10), KRDENO	OMNI 4
1, NEKCO(10, 31), KSAVE, NSAVE, NPLAO	OMNI 5
COMMON / BLOCKO / RC(2439), IARGO(69), KIND(39), ARGTAB(51), NGRMX,	OMNI 6
1 NROH, NCOL, NARGO, VHXZY(5)	OMNI 7
COMMON / QRG / NORON, JRON, NNARG	OMNI 8
COMMON / BLOCKE / NAME(4), L1, L2, ISRFLO	OMNI 9
COMMON KEY, IOVLY, ITYPE	OMNI 10
DIMENSION TEXT(2)	OMNI 11
	OMNI 12
	OMNI 13
THIS SUBROUTINE IS THE OMNITAB DRIVER SUBROUTINE.	OMNI 14
	OMNI 15
MODE = 1 - INTERPRETIVE MODE	OMNI 16
MODE = 2 - DATA MODE (READ AND SET)	OMNI 17
	OMNI 18
THE ARRAY KARO CONTAINS THE NUMERICAL REPRESENTATION OF THE INPUT	OMNI 19
CHARACTERS, AS FOLLOWS:	OMNI 20
0 = 0, 1 = 1, ETC., 9 = 9, A = 10, B = 11, ETC., Z = 35, / = 36	OMNI 21
. = 37, - = 38, + = 39, * = 40, (= 41,) = 42, . = 43	OMNI 22
BLANK = 44, = = 45, 0 AND OTHERS = 46	OMNI 23
	OMNI 24
NOUN=4	OMNI 25
KEY=-1	OMNI 26
MASKO=2147410110	OMNI 27
IOVLY = 1	OMNI 28
CALL GCEOD(EODINT)	OMNI 29
CALL OCPRK(MASKO, PPL, T)	OMNI 30
CALL DISPLY(450)	OMNI 31
50 NAME(1)=0	OMNI 32
NAME(2)=0	OMNI 33
NAME(3)=0	OMNI 34
NAME(4)=0	OMNI 35
NARGO=0	OMNI 36
J=0	OMNI 37
52 IF (KEY .EQ. 31) RETURN	OMNI 38
IF (NPLAO .EQ. 1 .OR. L1 .EQ. 0) GO TO 525	OMNI 39
IF (MODE .EQ. 2 .AND. JRON .LT. 00 .AND. ISRFLO .EQ. 0) GO TO 524	OMNI 40
CALL OCKSP(5)	OMNI 41
CALL OROPLY('READY', 0.4526)	OMNI 42
CALL OROPLY(' ', 1.4525)	OMNI 43
CALL OROPLY(' ', 1.4525)	OMNI 44
CALL OROPLY(' ', 1.4525)	OMNI 45
CALL OROPLY(' ', 1.4525)	OMNI 46
GO TO 526	OMNI 47
523 CALL OEROS(100)	OMNI 48
GO TO 5250	OMNI 49
524 JJ=JRON+1	OMNI 50
CALL OROPLY(' ', 1.4525)	OMNI 51
CALL OROPLY(0)	OMNI 52
5250 WRITE(NOUN, 998) JJ	OMNI 53
999 FORMAT('NOUN', 10, ' ')	OMNI 54

CALL FETCH(TEXT,NCF,4525)	OHNI 55
CALL GRAPLY(TEXT,NCF,4525)	OHNI 56
CALL GROPLY(' '.1,4525)	OHNI 57
CALL GROPLY(' '.1,4525)	OHNI 58
CALL GROPLY(' '.1,4525)	OHNI 59
CALL GROPLY(' '.1,4525)	OHNI 60
625 CALL WAIT	OHNI 61
IF(ITYPE.NE.1) GO TO 54	OHNI 62
IF(KEY.LT.4) GO TO 53	OHNI 63
IF(KEY.LT.10) CALL WORKD(4525)	OHNI 64
IF(KEY.NE.22) GO TO 535	OHNI 65
CALL SCOPT 10,IFRAME,KIND.SIZE,SPACE,INTEQ,IRCODE)	OHNI 66
CALL SCOPT (1,10UH)	OHNI 67
CALL GCALH	OHNI 68
GO TO 525	OHNI 69
53 IF(KEY.EQ.9) CALL CONAND(452)	OHNI 70
IF(KEY.EQ.2) CALL PRORAM(0)	OHNI 71
535 IF(KEY.EQ.90) CALL DISPLY(452)	OHNI 72
IF(KEY.EQ.31) RETURN	OHNI 73
GO TO 52	OHNI 74
54 CALL INPUT	OHNI 75
C SCANNING BEGINS WITH THE THIRD CHARACTER. THE FIRST TWO ARE DUMMY	OHNI 76
C TO KEEP THE PROGRAM OUT OF TROUBLE. SCANNING TERMINATES WITH A \$	OHNI 77
C A \$ HAS BEEN PLANTED IN THE (KRCEND+1)-TH POSITION.	OHNI 78
C	OHNI 79
NFLAG=0	OHNI 80
N=2	OHNI 81
55 N=N+1	OHNI 82
K=KAND(N)	OHNI 83
IF(K.OE.30)IF(K-40)55.50.55	OHNI 84
IF(K.OE.10)GO TO 70	OHNI 85
IF(MODE.EQ.2) GO TO 60	OHNI 86
CALL ERROR(7)	OHNI 87
GO TO 52	OHNI 88
C N IS POINTING AT THE FIRST LETTER ON THE CARD. ASSEMBLE NAME.	OHNI 89
C	OHNI 90
70 CALL NAME(NAME(1))	OHNI 91
C	OHNI 92
C	OHNI 93
C CHECK THE FIRST NAME FOR SPECIAL NAMES...	OHNI 94
C ONNITAD,STOP,ROW	OHNI 95
C	OHNI 96
C ONNITAD	OHNI 97
C	OHNI 98
C	OHNI 99
IF(NAME(1).NE.11305.OR.NAME(2).NE.7102100 TO 07	OHNI 100
CALL XNNIT	OHNI 101
GO TO 50	OHNI 102
C	OHNI 103
C	OHNI 104
C	OHNI 105
67 IF(NAME(1).NE.14406.OR.NAME(2).NE.11064100 TO 00	OHNI 106
RETURN	OHNI 107
C	OHNI 108

C	ROW	ORNI 108
C		ORNI 110
	88 IF(NAME(1).NE.19550.OR.NAME(2).NE.0) GO TO 89	ORNI 111
	IF(MODE.NE.2.OR.ISRFLG.NE.0) GO TO 888	ORNI 112
	894 K=KARD(M)	ORNI 113
	IF(K.GE.10) IF(K-40) 805,807,805	ORNI 114
	CALL ARRG	ORNI 115
	JROH=ARG-1.	ORNI 116
	GO TO 203	ORNI 117
	805 H=H+1	ORNI 118
	GO TO 834	ORNI 119
	806 CALL ERROR(7)	ORNI 120
	GO TO 52	ORNI 121
	807 CALL ERROR(10)	ORNI 122
	GO TO 525	ORNI 123
C		ORNI 124
C	H IS POINTING AT THE FIRST NON-LETTER AFTER NAME. LOOK FOR	ORNI 125
C	POSSIBLE NAME QUALIFIER OR ARGUMENTS OR END OF CARD.	ORNI 126
C		ORNI 127
	89 K=KARD(M)	ORNI 128
	IF(K.LT.36)IF(K-10)100,30,30	ORNI 129
	IF(K.EQ.40)GO TO 100	ORNI 130
	IF(K.EQ.46)GO TO 200	ORNI 131
	H=H+1	ORNI 132
	GO TO 88	ORNI 133
C		ORNI 134
C	A LETTER FOUND. ASSEMBLE SECOND NAME (COMMAND QUALIFIER).	ORNI 135
C		ORNI 136
C	80 CALL NAME(NAME(1))	ORNI 137
C		ORNI 138
C	CHECK SPECIAL CASE OF NAMES H(X)X', H(X)X), H(X)X', H(X)X)	ORNI 139
C		ORNI 140
C	SKIP ONE CHARACTER (') IF FIRST NAME = (H)	ORNI 141
C	IF(NAME(1) .EQ. 9477) H = H + 1	ORNI 142
	GO TO 100	ORNI 143
C		ORNI 144
C	SCAN FOR ARGUMENTS AND END OF CARD	ORNI 145
C		ORNI 146
	80 H=0	ORNI 147
	100 J=J+1	ORNI 148
	GO TO 102	ORNI 149
	101 H=H+1	ORNI 150
	102 K=KARD(M)	ORNI 151
	IF(K.GE.10)IF(K-40)101,120,103	ORNI 152
C		ORNI 153
C	NUMBER FOUND. CONVERT ARGUMENT. IF KARD RETURNED = 0. NUMBER IS	ORNI 154
C	INTEGER. IF KARD = 1. NUMBER IS FLOATING POINT. IF KARD = -1. ERROR	ORNI 155
C		ORNI 156
	CALL ARRG	ORNI 157
	IF(KARD)120,105,103	ORNI 158
	103 ARGARG(J)=0.	ORNI 159
	J=J+1	ORNI 160
	GO TO 110	ORNI 161
C		ORNI 162

C	ARGUMENT IS AN INTEGER. ADD A BIAS OF 8192 THEN CHECK THAT IT IS	OMNI 163
C	.GT. 0	OMNI 164
C		OMNI 165
105	ARG=ARG+8192.	OMNI 166
	IF(ARG.GT.0.)GO TO 110	OMNI 167
	CALL ERROR(18)	OMNI 168
	GO TO 50	OMNI 169
110	ARGTAB(J)=ARG	OMNI 170
115	NARG8 = NARG8 + 1	OMNI 171
	GO TO 100	OMNI 172
C		OMNI 173
C	ASTERISK FOUND. CONVERT	OMNI 174
C		OMNI 175
C	IF BRACKETED BY SINGLE ASTERISKS. QUANTITY IS TO BE USED AS A	OMNI 176
C	FLOATING POINT ARGUMENT. IF BRACKETED BY DOUBLE ASTERISKS. QUANTITY	OMNI 177
C	IS TO BE TRUNCATED AND USED AS AN INTEGER ARGUMENT.	OMNI 178
C		OMNI 179
120	KARG=1	OMNI 180
	M=M+1	OMNI 181
	IF(KARG(M).NE.40)GO TO 125	OMNI 182
	KARG=0	OMNI 183
	M=M+1	OMNI 184
125	CALL ASTER	OMNI 185
C		OMNI 186
C	THE TERMINAL ASTERISK(6) HAVE BEEN CHECKED TO BE THE SAME AS THE	OMNI 187
C	INITIAL SET (IF NO ERROR) AND N IS POINTING AT THE FIRST CHARACTER	OMNI 188
C	AFTER THE LAST ASTERISK.	OMNI 189
C		OMNI 190
C	KARG RETURNED AS 1 = ERROR FOUND	OMNI 191
C	2 = FLOATING POINT CONSTANT. Z.B. 3.14159	OMNI 192
C	3 = INTEGER NAMED VARIABLE. Z.B. MAXX	OMNI 193
C	4 = FL. PT. NAMED VARIABLE. Z.B. MAXX	OMNI 194
C	5 = INTEGER ROW-COLUMN. Z.B. 3,20	OMNI 195
C	6 = FL. PT. ROW-COLUMN. Z.B. 3.20	OMNI 196
C	7 = STRING OF ASTERISKS Z.B. ***	OMNI 197
C		OMNI 198
C	A STRING OF THREE OR MORE ASTERISKS IMPLIED -THRU-	OMNI 199
C	EXAMPLE..	OMNI 200
C	ERASE 1 2 3 4 12 13 14 15 16 20 IS EQUIVALENT TO	OMNI 201
C	ERASE 1 thru 4. 12 thru 16. 20	OMNI 202
C		OMNI 203
	GO TO (50, 100, 135, 136, 140, 140, 150), KARG	OMNI 204
135	ARGTAB(J)=2.0*ARG-FLOAT(KARG-3)	OMNI 205
	GO TO 115	OMNI 206
140	ARGTAB(J)=ARG-8200.	OMNI 207
	ARG2=ARG2+8192.	OMNI 208
	IF(KARG.EQ.6)ARG2=ARG2	OMNI 209
	J=J+1	OMNI 210
	ARGTAB(J)=ARG2	OMNI 211
	GO TO 115	OMNI 212
150	IF(J .GT. 1) GO TO 155	OMNI 213
	CALL ERROR(211)	OMNI 214
	GO TO 102	OMNI 215
155	ARGTAB(J) = -1.	OMNI 216

KSAVE=KSAVE+1	ORNI 271
DO 9002 JJJ=1,19	ORNI 272
9002 NENCD(JJJ,KSAVE)=NENCD(JJJ)	ORNI 273
IF(KSAVE.LT.5) GO TO 50	ORNI 274
IF(IJVLV.GT.40) GO TO 9008	ORNI 275
9005 WRITE(KSAVE,IJVLV) NENCD	ORNI 276
KSAVE=0	ORNI 277
GO TO 50	ORNI 278
9006 CALL PROGRAM(1)	ORNI 279
IF(KEY.EQ.31) RETURN	ORNI 280
IJVLV = 1	ORNI 281
GO TO 9005	ORNI 282
C	ORNI 283
C	ORNI 284
C	ORNI 285
C	ORNI 286
210 CALL LOOKUP	ORNI 287
IF(L1.NE.0) GO TO 220	ORNI 288
IF(MODE.EQ.2) GO TO 202	ORNI 289
CALL ERROR(1)	ORNI 290
GO TO 00	ORNI 291
C	ORNI 292
C	ORNI 293
C	ORNI 294
220 MODE=1	ORNI 295
222 CALL EXPAND(J,ARGTAB)	ORNI 296
CALL EXECUTE	ORNI 297
GO TO 60	ORNI 298
END	ORNI 299

	SUBROUTINE PROMOTE_____	POMO 1
	COMMON / BLOCKA/NDDE,M,KARD(77),KARG,ARD,ARG2,NENCO(19),KROEND	POMO 2
	1,NEVCOS(19,5),KSAVE,NSAVE,NELAD	POMO 3
	COMMON / BLOCKB / NC(2439),IARGOS(69),KIND(39),AROTAB(51),NRMAX	POMO 4
	1,NROW,NCOL,NARGOS,VWXYZ(5)	POMO 5
	COMMON/BLOCKC/NAHE(4),L1,L2,ISRFLD	POMO 6
C		POMO 7
C	THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS	POMO 8
C	PROMOTE AND DEMOTE.	POMO 9
C	L2 = 10 (OR 0) - PROMOTE	POMO 10
C	L2 = 11 (OR 1) - DEMOTE	POMO 11
C		POMO 12
	L2 = L2 - 10	POMO 13
	IF(NDCI NARGOS, 2 1, NE, 0) GO TO 30	POMO 14
	I = 10	POMO 15
10	CALL ERROR(1)	POMO 16
20	RETURN	POMO 17
30	NR = IARGOS(1)	POMO 18
	NARGOS=NARGOS-I	POMO 19
	IF(NARGOS.EQ.0) GO TO 40	POMO 20
	DO 31 I=1,NARGOS	POMO 21
31	IARGOS(I)=IARGOS(I+1)	POMO 22
	IF(NRMAX.GT.0) GO TO 32	POMO 23
	I=9	POMO 24
	GO TO 10	POMO 25
32	CALL CHKCOL(I)	POMO 26
	IF(I .EQ. 0) GO TO 40	POMO 27
35	I = 9	POMO 28
	GO TO 10	POMO 29
C		POMO 30
C	IF NUMBER OF ROWS TO BE MOVED IS NEGATIVE, FLIP INSTRUCTIONS.	POMO 31
C	I.E. PROMOTE -G IS THE SAME AS DEMOTE G	POMO 32
C		POMO 33
	40 IF(NR.EQ.0) GO TO 60	POMO 34
	L2 = 1 - L2	POMO 35
	NR = -NR	POMO 36
C		POMO 37
C	CHECK DISTANCE OF MOVE	POMO 38
C		POMO 39
	60 IF(L2 .EQ. 0) GO TO 80	POMO 40
	IF(NR + NRMAX .LE. NROW) GO TO 100	POMO 41
	GO TO 35	POMO 42
	80 IF(NR - NRMAX) 100, 90, 35	POMO 43
C		POMO 44
C	PROMOTE 'NRMAX' ...	POMO 45
C		POMO 46
	90 CALL PICK	POMO 47
	IF(NRFLD.EQ.1.OR.NARGOS.NE.0) GO TO 20	POMO 48
	J=1	POMO 49
	DO 95 I = 1, NCOL	POMO 50
	CALL VECTOR(0., J)	POMO 51
95	J = J + NROW	POMO 52
	GO TO 20	POMO 53
100	LIMIT = NARGOS	POMO 54

```

      IF( LIMIT .EQ. 0 ) LIMIT = 2 * NCOL
      CALL PLUX
      IF( NPLAO .EQ. 1 ) GO TO 20
C
C      START PROMOTING OR DEMOTING
C
110 KK=1
      DO 200 I=1,LIMIT,2
      IF( NAROS .NE. 0 ) GO TO 120
      K1=KK
      K2 = K1
      KK=KK+NRON
      GO TO 130
120 K1=IAROS(I)
      K2=IAROS(I+1)
130 IF( L2 .EQ. 0 ) GO TO 150
C
C      DEMOTE COL AT K1 TO COL AT K2
C
      K1 = K1 + NRMAX
      K2 = K2 + NRMAX + NR
      DO 140 J = 1, NRMAX
      K1 = K1 - 1
      K2 = K2 - 1
140 RC( K2 ) = RC( K1 )
      GO TO 200
C
C      PROMOTE COL AT K1 TO COL AT K2
C
150 JJ = NRMAX - NR
      K1 = K1 + NR
      DO 160 J = 1, JJ
      RC( K2 ) = RC( K1 )
      K1 = K1 + 1
      K2 = K2 + 1
160
C
C      IF PROMOTE ARRAY, FILL REST OF COLUMN WITH ZEROS.
C
      IF( NAROS .NE. 0 ) GO TO 200
      JJ = JJ + 1
      DO 170 J = JJ, NRMAX
      RC( K2 ) = 0.
170 K2 = K2 + 1
200 CONTINUE
      IF( L2 .NE. 0 ) NRMAX = NRMAX + NR
      GO TO 20
      END

```

```

PDHO 55
PDHO 56
PDHO 57
PDHO 58
PDHO 59
PDHO 60
PDHO 61
PDHO 62
PDHO 63
PDHO 64
PDHO 65
PDHO 66
PDHO 67
PDHO 68
PDHO 69
PDHO 70
PDHO 71
PDHO 72
PDHO 73
PDHO 74
PDHO 75
PDHO 76
PDHO 77
PDHO 78
PDHO 79
PDHO 80
PDHO 81
PDHO 82
PDHO 83
PDHO 84
PDHO 85
PDHO 86
PDHO 87
PDHO 88
PDHO 89
PDHO 90
PDHO 91
PDHO 92
PDHO 93
PDHO 94
PDHO 95
PDHO 96
PDHO 97
PDHO 98
PDHO 99
PDHO 100
PDHO 101

```

SUBROUTINE PFIN(ARQ)	PFIN	1
COMMON KEY,JOVLY,ITYPE	PFIN	2
INTEGER*2 KKEY(2), AREA=4(5)	PFIN	3
EQUIVALENCE (NA,KKEY(1))	PFIN	4
NA = AREA(1)	PFIN	5
KEY = KKEY(2)/256	PFIN	6
ITYPE=1	PFIN	7
CALL OPOST	PFIN	8
RETURN	PFIN	8
END	PFIN	10
SUBROUTINE PHYCON(NAHE)	PHYC	1
COMMON / BLOCKN/HODE,II,KARD(77),KARG,ARG,ARG2,NENCO(19),KRDEND	PHYC	2
1,NENCO(19,5),KSAVE,NSAVE,NFLAG	PHYC	3
COMMON/PCONST/P(2),N(2)	PHYC	4
	PHYC	5
THIS SUBROUTINE IS USED TO DETERMINE IF A PHYSICAL CONSTANT IS	PHYC	6
STORED IN THE SYSTEM. IF SO IT RETURNS IT IN ARG.	PHYC	7
	PHYC	8
DO 20 IH=1,2	PHYC	9
I = IH	PHYC	10
IF(NAHE.EQ.N(1))GO TO 30	PHYC	11
20 CONTINUE	PHYC	12
ARG=0.	PHYC	13
RETURN	PHYC	14
30 ARG=P(1)	PHYC	15
RETURN	PHYC	16
END	PHYC	17

SUBROUTINE PROGRAM(101)	PRGR 1
COMMON / BLOCKA/MODE,M,KARD(77),KARG,ARG,ARG2,NEWCD(19),KRDEND	PRGR 2
1,NEWCD(19,5),KSAVE,HSAVE,HFLAG	PRGR 3
COMMON/KPLOT/NFRAME,KKNO,SIZE,SPACE	PRGR 4
COMMON KEY,IOVLY,ITYPE	PRGR 5
DIMENSION NTEXT(95),NT(19)	PRGR 6
	PRGR 7
THIS SUBROUTINE IS USED TO DISPLAY THE COMMANDS WHICH HAVE BEEN	PRGR 8
EXECUTED.	PRGR 9
	PRGR 10
IF(101.EQ.0) GO TO 0	PRGR 11
CALL GROPY(' ',1,41000)	PRGR 12
CALL GROPY('AVAILABLE SPACE FOR STORING COMMANDS HAS BEEN FILLED.	PRGR 13
1 IF YOU WISH TO SEE',73,41000)	PRGR 14
CALL GROPY('THE COMMANDS WHICH YOU HAVE ENTERED BEFORE THEY ARE	PRGR 15
ERASED, PRESS KEY 2.',72,41000)	PRGR 16
CALL GROPY('OTHERWISE, PRESS KEY 1.',23,41000)	PRGR 17
2 CALL GCOLH	PRGR 18
25 CALL ORHIT	PRGR 19
IF(ITYPE.NE.1.OR.KEY.NE.2) GO TO 2	PRGR 20
IF(KEY.EQ.1.OR.KEY.EQ.3) RETURN	PRGR 21
IF(KEY.EQ.2) GO TO 0	PRGR 22
IF(KEY.NE.22) GO TO 2	PRGR 23
CALL GCOPLY(0,NFRAME,KKNO,SIZE,SPACE,INTEQ,IRCODE)	PRGR 24
CALL GCOPLY(1,IOVLY)	PRGR 25
CALL GCOLH	PRGR 26
GO TO 25	PRGR 27
0 CALL ORHIS(100)	PRGR 28
CALL GROPY('IF THE SCREEN BECOMES FULL AN ALARM WILL SOUND. WHEN	PRGR 29
1 YOU WANT TO SEE',69,41000)	PRGR 30
CALL GROPY('THE NEXT SECTION OF YOUR PROGRAM, PRESS KEY 2.',40,41	PRGR 31
1000)	PRGR 32
CALL GROPY(' ',1,41000)	PRGR 33
IF(IOVLY.EQ.1) GO TO 30	PRGR 34
J=IOVLY-1	PRGR 35
IOVLY=1	PRGR 36
DO 50 M=1,J	PRGR 37
READ(NSAVE(IOVLY)) NTEXT	PRGR 38
DO 50 I=1,99,10	PRGR 39
40 CALL GROPY(NTEXT(I),72,4100)	PRGR 40
GO TO 50	PRGR 41
150 CALL GCOLH	PRGR 42
160 CALL ORHIT	PRGR 43
IF(ITYPE.NE.1) GO TO 150	PRGR 44
IF(KEY.EQ.3) RETURN	PRGR 45
IF(KEY.EQ.2) GO TO 45	PRGR 46
IF(KEY.NE.22) GO TO 150	PRGR 47
CALL GCOPLY(0,NFRAME,KKNO,SIZE,SPACE,INTEQ,IRCODE)	PRGR 48
CALL GCOPLY(1,IOVLY)	PRGR 49
CALL GCOLH	PRGR 50
GO TO 100	PRGR 51
45 CALL ORHIS(100)	PRGR 52
GO TO 40	PRGR 53
50 CONTINUE	PRGR 54

```

90 IF(KGAVE.EQ.0) GO TO 400
   DO 110 I=1,NSAVE
   DO 100 J=1,19
100 NTIJ=NEUCOS(J,I)
240 CALL GROPY(NT,72,4250)
   GO TO 110
250 CALL CCALN
260 CALL CHAIT
   IF(ITYPE.NE.1) GO TO 250
   IF(KEY.EQ.31) RETURN
   IF(KEY.EQ.2) GO TO 245
   IF(KEY.NE.22) GO TO 250
   CALL SCOPLT(0,NFRAME,KIND,SIZE,SPACE,INTEQ,IRCODE)
   CALL SCOPLT(1,ICUM)
   CALL CCALN
   GO TO 260
245 CALL GENAS(100)
   GO TO 240
110 CONTINUE
400 CALL GROPY(' '.1,41000)
   CALL GROPY(' '.1,41000)
   CALL GROPY(' '.1,41000)
   CALL GROPY(' '.1,41000)
   CALL GROPY(' '.1,41000)
1000 RETURN
      END

```

```

PRGR 55
PRGR 56
PRGR 57
PRGR 58
PRGR 59
PRGR 60
PRGR 61
PRGR 62
PRGR 63
PRGR 64
PRGR 65
PRGR 66
PRGR 67
PRGR 68
PRGR 69
PRGR 70
PRGR 71
PRGR 72
PRGR 73
PRGR 74
PRGR 75
PRGR 76
PRGR 77
PRGR 78
PRGR 79
PRGR 80

```

	SUBROUTINE PROSON	PROR	1
	COMMON / BLOCKA/MODE,N,KARD(77),KARG,ARG,ARG2,NEHCO(19),KRDEHD	PROR	2
	1,NEHCO(19,5),KSAVE,NSAVE,NFLAO	PROR	3
	COMMON / BLOCKB / RC(2499),IARG(89),KIND(39),AROTAS(51),NRMAX,	PROR	4
	1 NRON,NCOL,NARCS,VWXYZ(5)	PROR	5
	COMMON/SCRAT/A(80)	PROR	6
	COMMON/BLOCKC/NAME(4),L1,L2,ISRFLD	PROR	7
	EQUIVALENCE (IA1,IARG(1)),(IA2,IARG(2))	PROR	8
C		PROR	9
C	THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS	PROR	10
C	ROMSUM AND PRODUCT.	PROR	11
C		PROR	12
	IF(NARCS.GE.3) GO TO 40	PROR	13
	CALL ERROR(10)	PROR	14
	RETURN	PROR	15
40	CALL CHKCOL(J)	PROR	16
	IF(J.EQ.0.AND.((IARG(2).GE.IARG(1).OR.NARCS.GT.9)) GO TO 60	PROR	17
50	CALL ERROR(13)	PROR	18
	RETURN	PROR	19
60	IF(NRMAX.GT.0) GO TO 80	PROR	20
	CALL ERROR(19)	PROR	21
	RETURN	PROR	22
80	CALL PLAK	PROR	23
	IF(NFLAO.EQ.1) RETURN	PROR	24
	CONST=0.	PROR	25
	IF(L2.EQ.2) CONST=1.	PROR	26
	DO 100 I=1,NRMAX	PROR	27
100	A(I)=CONST	PROR	28
	IF(NARCS.GE.4) GO TO 200	PROR	29
	DO 150 K=IA1,IA2,NRON	PROR	30
	DO 150 I=1,NRMAX	PROR	31
	J = K + I - 1	PROR	32
	IF(L2.EQ.2) GO TO 140	PROR	33
	A(I)=A(I)+RC(J)	PROR	34
	GO TO 150	PROR	35
140	A(I) = A(I)+RC(J)	PROR	36
150	CONTINUE	PROR	37
170	K = IARG(NARCS)	PROR	38
	DO 180 I=1,NRMAX	PROR	39
	J = K + I - 1	PROR	40
180	RC(J) = A(I)	PROR	41
	RETURN	PROR	42
200	II = NARCS - 1	PROR	43
	DO 250 L=1,II	PROR	44
	K = IARG(L)	PROR	45
	DO 250 I=1,NRMAX	PROR	46
	J = K + I - 1	PROR	47
	IF(L2.EQ.2) GO TO 240	PROR	48
	A(I)=A(I)+RC(J)	PROR	49
	GO TO 250	PROR	50
240	A(I) = A(I)+RC(J)	PROR	51
250	CONTINUE	PROR	52
	GO TO 170	PROR	53
	END	PROR	54

	SUBROUTINE READQ	READ	1
	COMMON / BLOCKA/MODE,N,KARD(77),KARO,ARO,ARG2,NEHCO(19),KROENO	READ	2
	1,NEHCO6(19,5),KSAVE,N6AVE,NFLAG	READ	3
	COMMON / BLOCKD / RC(2499),IARCS(69),KIND(99),AROTAB(51),NRMAX,	READ	4
	1 NRON,NCOL,NAROS,VNXYZ(5)	READ	5
	DIMENSION ARCS(39)	READ	6
	EQUIVALENCE(ARCS(1),RC(2401))	READ	7
	COMMON/ARG/NORON,J,NNARO	READ	8
		READ	9
	THIS SUBROUTINE IS USED TO PUT A ROW OF DATA INTO THE WORKSHEET.	READ	10
	IF(J .LT. NRON) GO TO 10	READ	11
	CALL ERROR(18)	READ	12
	GO TO 99	READ	13
		READ	14
		READ	15
	NNAROS CONTAINS THE NUMBER OF ARGUMENTS IN THE READ COMMAND.	READ	16
	IARCS(40) THROUGH IARCS(NNARO-39) CONTAIN ADDRESSES OF COLUMNS.	READ	17
		READ	18
10	DO 30 I = 1, NNARO	READ	19
	K = IARCS(I + 39) + J	READ	20
	IF(KIND(I) .EQ. 0) GO TO 20	READ	21
	RC(K) = ARCS(I)	READ	22
	GO TO 30	READ	23
20	RC(K) = IARCS(I)	READ	24
30	CONTINUE	READ	25
	CALL GUKSP(4)	READ	26
	CALL GROPY(NEHCO,70,499)	READ	27
		READ	28
	J IS THE NUMBER OF ROWS READ IN.	READ	29
		READ	30
	J = J + 1	READ	31
	IF(NRMAX .LT. J) NRMAX = J	READ	32
	CALL GROPY(' ',1,499)	READ	33
	CALL GROPY(' ',1,499)	READ	34
	CALL GROPY(' ',1,499)	READ	35
	CALL GROPY(' ',1,499)	READ	36
	CALL GROPY(' ',1,499)	READ	37
99	RETURN	READ	38
	END	READ	39

SUBROUTINE READX	READ 1
COMMON / BLOCKN/MODE,M,KARD(77),KARG,ARG,ARGZ,NEWCD(19),KROEND	READ 2
1,NEWCDG(19,5),KSAVE,NSAVE,HFLAG	READ 3
COMMON / BLOCKD / RC(2499),IAROS(69),KIND(99),AROTAB(51),NRMAX,	READ 4
1 NRON,NCOL,NAROS,VWXYZ(5)	READ 5
DIMENSION AROS(99)	READ 6
COMMON/BLOCKE/NAME(4),L1,L2,ISRFLG	READ 7
COMMON/ARS/NORON,J,NHARG	READ 8
EQUIVALENCE(AROS(1),RC(2401))	READ 9
C	READ 10
C THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMAND READ.	READ 11
C	READ 12
IF(NAROS .GT. 0) GO TO 10	READ 13
5 CALL ERROR(10)	READ 14
GO TO 100	READ 15
10 CALL CHKCOL(1)	READ 16
IF(1 .EQ. 0) GO TO 20	READ 17
15 CALL ERROR(9)	READ 18
GO TO 100	READ 19
201 CALL CERAS(100)	READ 20
GO TO 21	READ 21
20 CALL OROPLY(' ',1,4201)	READ 22
21 CALL OOKSP(6)	READ 23
MODE=2	READ 24
CALL OROPLY(NEWCD,70,4100)	READ 25
CALL OROPLY(' ',1,4100)	READ 26
CALL OROPLY(' ',1,4100)	READ 27
CALL OROPLY(' ',1,4100)	READ 28
CALL OROPLY(' ',1,4100)	READ 29
CALL OROPLY(' ',1,4200)	READ 30
200 ISRFLG = 0	READ 31
DO 30 I = 1, NAROS	READ 32
IAROS(I + 99) = IAROS(I)	READ 33
IAROS(I) = 0	READ 34
30 AROS(I) = 0.	READ 35
J = 0.	READ 36
NHARG = NAROS	READ 37
100 RETURN	READ 38
END	READ 39

	SUBROUTINE RESET	RESE	1
	COMMON / BLOCKA/NODE.H,KARO(77),KARO,ARG,ARG2,NENCO(10),KROEND	RESE	2
	1,NENCOS(10,5),KSAVE,NSAVE,NFLAG	RESE	3
	COMMON / BLOCKD / RC(2400),IARGS(60),KIND(90),ARGTAB(51),NRMAX,	RESE	4
	1 NRON,NCOL,NAROS,VWXYZ(6)	RESE	5
	DIMENSION ARS(1)	RESE	6
	COMMON/BLOCKE/NAME(4),L1,L2,ISRFLO	RESE	7
	EQUIVALENCE(ARS(1),RC(2401))	RESE	8
		RESE	9
	THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMAND RESET.	RESE	10
	IF (NAROS .EQ. 1) IF (L2 - 5)100,100,30	RESE	11
	K = 10	RESE	12
10	CALL ERROR(K)	RESE	13
20	RETURN	RESE	14
		RESE	15
	RESET NRMAX	RESE	16
		RESE	17
30	IF(KIND(1) .NE. 0) IARGS(1) = ARS(1)	RESE	18
40	IF(IARGS(1) .GE. 0 .AND. IARGS(1) .LE. NRON) GO TO 30	RESE	19
	K = 9	RESE	20
	GO TO 10	RESE	21
50	CALL PLSK	RESE	22
	IF(NFLAG.EQ.1) RETURN	RESE	23
	NRMAX = IARGS(1)	RESE	24
	GO TO 20	RESE	25
		RESE	26
	RESET V.N.X.Y.Z	RESE	27
		RESE	28
100	CALL PLSK	RESE	29
	IF(NFLAG.EQ.1) RETURN	RESE	30
	IF(KIND(1) .EQ. 0) ARS(1) = IARGS(1)	RESE	31
	VWXYZ(L2) = ARS(1)	RESE	32
	GO TO 20	RESE	33
	END	RESE	34
	SUBROUTINE SCRA(NC,IT)	RESE	35
	COMMON / BLOCKA/NODE.H,KARO(77),KARO,ARG,ARG2,NENCO(10),KROEND	SCRA	1
	1,NENCOS(10,5),KSAVE,NSAVE,NFLAG	SCRA	2
	COMMON/SCRA/RC(20)	SCRA	3
	COMMON KEY,IOVLY,ITYPE	SCRA	4
	IPTR=IOVLY	SCRA	5
	NCOL=NC+40	SCRA	6
	IF(IT.EQ.2) GO TO 50	SCRA	7
	READ(NSAVE'NCOL) A	SCRA	8
	GO TO 100	SCRA	9
50	WRITE(NSAVE'NCOL) A	SCRA	10
100	IOVLY=IPTR	SCRA	11
	RETURN	SCRA	12
	END	SCRA	13
		SCRA	14

	SUBROUTINE SET	GET	1
	COMMON / BLOCKA/MODE,H,KARO(77),KARG,ARG,ARG2,NEHCO(19),KROEND	SET	2
	1,NEHCO6(19,5),KSAVE,NSAVE,NFLAO	SET	3
	COMMON / BLOCKD / RC(2493),IARG(63),KIND(93),AROTAB(51),NRHAX,	SET	4
	1 NRON,NCOL,NAROS,VHXYE(5)	SET	5
	COMMON/BLOCKE/NAME(4),L1,L2,I6RFLO	SET	6
	COMMON/GRS/NORON,J,NNRRO	SET	7
C		SET	8
C	THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMAND SET.	SET	9
C		SET	10
	IF (NAROS .GE. 1 .AND. NAROS .LT. 3) GO TO 10	SET	11
	CALL ERROR(10)	SET	12
	GO TO 100	SET	13
10	CALL ADDRESS(NAROS , J)	SET	14
	IF(J) 15, 17, 20	SET	15
15	CALL ERROR(9)	SET	16
	GO TO 100	SET	17
17	CALL ERROR(11)	SET	18
	GO TO 100	SET	19
20	NORON = J + NRON - 1	SET	20
	IF(NAROS .EQ. 1) GO TO 25	SET	21
	IF(KIND(1) .NE. 0) GO TO 15	SET	22
	IF(IAROS(1) .LE. NRON .AND. IAROS(1) .GT. 0) GO TO 24	SET	23
	CALL ERROR(16)	SET	24
	GO TO 100	SET	25
24	J = J + IAROS(1) - 1	SET	26
25	CALL PLAK	SET	27
	IF(INFLAG.EQ.1) RETURN	SET	28
	ISRFLO = 1	SET	29
	MODE = 2	SET	30
100	RETURN	SET	31
	END	SET	32

	SUBROUTINE SETQ	SETQ 1
	COMMON / CLOCKR/MODE.N,KARD(77),KARG,ARG,ARG2,NEXC0(10),KROEND	SETQ 2
	1,NEXC0S(10,5),KSAVE,NSAVE,NFLAG	SETQ 3
	COMMON / CLOCKD / RC(2400),IAROS(60),KIND(90),ARGTAB(51),NRMAX,	SETQ 4
	1 NR0N,NCOL,NAROS,VWXYZ(5)	SETQ 5
	DIMENSION ARGS(90)	SETQ 6
	EQUIVALENCE(ARGS(1),RC(2401))	SETQ 7
	COMMON/ARS/NOROW,J,NKARD	SETQ 8
		SETQ 9
C	THIS SUBROUTINE IS USED TO ENTER DATA INTO THE WORKSHEET	SETQ 10
C	IN RESPONSE TO THE SET COMMAND.	SETQ 11
C	JJ IS WHERE NEXT DATA ITEM IS TO GO IN COLUMN	SETQ 12
C	JJ IS WHERE LAST DATA ITEM OF THIS SET IS TO GO	SETQ 13
C	NOROW IS ADDRESS OF LAST ROW IN COLUMN.	SETQ 14
C		SETQ 15
	IF(NAROS.EQ.0) GO TO 99	SETQ 16
	JJ = J + NAROS - 1	SETQ 17
	IF(JJ .LE. NOROW) GO TO 10	SETQ 18
	CALL ERROR(201)	SETQ 19
	IF(NFLAG.EQ.1) RETURN	SETQ 20
C		SETQ 21
C	CHECK IF END OF ROW HAS BEEN EXCEEDED PREVIOUSLY IN THIS SET.	SETQ 22
C		SETQ 23
	IF(J .GT. NOROW) GO TO 99	SETQ 24
	JJ = NOROW	SETQ 25
	GO TO 15	SETQ 26
10	CALL PLSK	SETQ 27
	IF(NFLAG.EQ.1) RETURN	SETQ 28
15	K = 1	SETQ 29
	DO 30 I = J, JJ	SETQ 30
	IF(KIND(K) .EQ. 0) GO TO 20	SETQ 31
	RC(I) = ARGS(K)	SETQ 32
	GO TO 30	SETQ 33
20	RC(I) = IAROS(K)	SETQ 34
30	K = K + 1	SETQ 35
	J = JJ + 1	SETQ 36
	NRMAX = MAX0(NRMAX, JJ - NOROW + NR0N)	SETQ 37
99	RETURN	SETQ 38
	END	SETQ 39

SUBROUTINE SORDER	6000	1
COMMON / BLOCKA/NDDE,N,KARD(77),KARD,ARG,ARG2,NEHCD(191,KADENO	6000	2
1,NEHCDG(10,5),KSAVE,NSAVE,NFLAG	6000	3
COMMON / BLOCKD / RC(2499),IARCS(69),KIND(99),ARCTAB(51),NRMAX,	6000	4
1 NRON,NCOL,NARGG,VWXYZ(5)	6000	5
COMMON/BLOCKE/NAME(4),L1,L2,ISRFLO	6000	6
COMMON/SCRAT/A(00)	6000	7
DIMENSION NUH(80)	6000	8
C	6000	9
C THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS	6000	10
C SORT, ORDER AND HIERARCHY.	6000	11
C L2=8 FOR SORT, L2=9 FOR ORDER, L2=14 FOR HIERARCHY	6000	12
C	6000	13
IF(NARGG.GT.0) GO TO 40.	6000	14
10 K=10	6000	15
20 CALL ERROR(K)	6000	16
30 RETURN	6000	17
40 CALL CHKCOL(J)	6000	18
IF(J.EQ.0) GO TO 80	6000	19
50 K=9	6000	20
GO TO 20	6000	21
60 IF(L2.NE.14) GO TO 80.	6000	22
IF(NARGG.NE.2) GO TO 10.	6000	23
80 IF(NRMAX.GT.0) GO TO 85	6000	24
K=9	6000	25
GO TO 20	6000	26
85 CALL PLUK	6000	27
IF(NFLAG.EQ.1) RETURN	6000	28
IF(NRMAX.GT.1) GO TO 120	6000	29
IF(L2.NE.14) GO TO 30	6000	30
RC(IARCS(2))=1.	6000	31
RETURN	6000	32
120 K3=1	6000	33
K = IARCS(1) -1	6000	34
130 DO 140 I =1,NRMAX	6000	35
J=K+1	6000	36
A(I) = RC(J)	6000	37
140 NUH(I)=1	6000	38
K1 = NRMAX	6000	39
160 K1 = K1 -1	6000	40
K2=0	6000	41
IF(K1.EQ.0) GO TO 210	6000	42
DO 200 I=1,K1	6000	43
IF(A(I).LE.A(I+1)) GO TO 200	6000	44
CC = A(I)	6000	45
A(I) = A(I+1)	6000	46
A(I+1) = CC	6000	47
ICC=NUH(I)	6000	48
NUH(I)=NUH(I+1)	6000	49
NUH(I+1)=ICC	6000	50
K2=1.	6000	51
200 CONTINUE	6000	52
IF(K2.EQ.1) GO TO 180	6000	53
210 IF(L2.NE.14) GO TO 240	6000	54

```

      K = IAROS(2) - 1
      DO 230 I=1,NRMAX
      J = K + 1
230  RC(I,J) = NUM(I)
      GO TO 30
240  DO 250 I=1,NRMAX
      J = K + 1
250  RC(I,J) = A(I)
      IF(NAROS.EQ.1) GO TO 30
      IF(L2.EQ.8) GO TO 290
      IF(NAROS.EQ.43) GO TO 30
      K3 = K3 + 1
      K = IAROS(K3) - 1
      DO TO 130
290  DO 310 I = 2,NAROS
      K = IAROS(I) - 1
      DO 300 J=1,NRMAX
      J1 = NUM(I) + K
300  A(J) = RC(I,J)
      DO 310 J=1,NRMAX
      J1 = K + J
310  RC(I,J) = A(J)
      GO TO 30
      END

```

```

60RD 55
60RD 56
60RD 57
60RD 58
60RD 59
60RD 60
60RD 61
60RD 62
60RD 63
60RD 64
60RD 65
60RD 66
60RD 67
60RD 68
60RD 69
60RD 70
60RD 71
60RD 72
60RD 73
60RD 74
60RD 75
60RD 76
60RD 77
60RD 78

```

SUBROUTINE SPINV(N,DET)	SPIN 1
COMMON/SCRAT/BI(J)	SPIN 2
REAL*8 A(40),DET,ONE,ZERO,ER,C,X,HOLD(40),Z,ZZ	SPIN 3
EQUIVALENCE (A,J)	SPIN 4
DATA ONE/1.00/,ZERO/0.00/,ER/1.0-4/	SPIN 5
C	SPIN 6
C THIS SUBROUTINE FINDS AN INVERSE BY USING THE GAUS-JORDAN METHOD OF	SPIN 7
C ELIMINATION. N WILL CONTAIN THE DIMENSION OF THE MATRIX TO BE	SPIN 8
C INVERTED. DET IS USED TO INDICATE WHETHER INVERSION WAS SUCCESSFUL.	SPIN 9
C DET = 0 - MATRIX WAS SINGULAR.	SPIN 10
C	SPIN 11
DET=ONE	SPIN 12
N = N	SPIN 13
N2 = N + N	SPIN 14
L = 0	SPIN 15
12 L = L + 1	SPIN 16
CALL SCRAN(L,1)	SPIN 17
C	SPIN 18
C FIND THE LARGEST ELEMENT IN THE LTH COLUMN.	SPIN 19
C	SPIN 20
J1 = L	SPIN 21
C=CRAS(A(L,1))	SPIN 22
IF(L - N)13,30,1000	SPIN 23
13 L1 = L + 1	SPIN 24
DO 20 J = L1,N	SPIN 25
CALL SCRAN(J,1)	SPIN 26
X=CRAS(A(L,1))	SPIN 27
IF(C.GE.X) DO TO 20	SPIN 28
C	SPIN 29
C RECORD THE NUMBER OF THE ROW HAVING THE GREATER ELEMENT.	SPIN 30
C	SPIN 31
J1 = J	SPIN 32
C	SPIN 33
C C BECOMES THE GREATER.	SPIN 34
C	SPIN 35
C = X	SPIN 36
20 CONTINUE	SPIN 37
C	SPIN 38
C IF THE LARGEST ELEMENT IN A COLUMN IS ZERO THERE IS A SINGULARITY.	SPIN 39
C	SPIN 40
30 IF(C.NE.ZERO) GO TO 22	SPIN 41
DET=ZERO	SPIN 42
GO TO 1000	SPIN 43
C	SPIN 44
C INTERCHANGE ROW J1 WITH ROW L. J1 IS THE ROW WITH THE LARGEST ELEMENT	SPIN 45
C TEST TO SEE IF INTERCHANGING IS NECESSARY.	SPIN 46
C	SPIN 47
72 IF(J1.EQ.L) GO TO 32	SPIN 48
CALL SCRAN(J1,1)	SPIN 49
DO 24 J = L,N2	SPIN 50
24 HOLD(J)=A(J)	SPIN 51
CALL SCRAN(L,1)	SPIN 52
CALL SCRAN(J1,2)	SPIN 53
DO 25 J=L,N2	SPIN 54

35 A(J)=HOLD(J)	6PIN 55
CALL SCRAH(L,2)	6PIN 56
C	6PIN 57
C ZERO ALL THE ELEMENTS IN THE LTH COLUMN BUT THE PIVOTAL ELEMENT.	6PIN 58
C	6PIN 59
32 L1 = 1	6PIN 60
L2 = L - 1	6PIN 61
L3=L+1	6PIN 62
CALL SCRAH(L,1)	6PIN 63
DO 3235 I=L,N2	6PIN 64
3235 HOLD(I)=A(I)	SPIN 65
IF(L2.GT.0) GO TO 323	SPIN 66
321 IF(L.EQ.N) GO TO 46	6PIN 67
322 L1=L3	SPIN 68
L2 = N	6PIN 69
323 DO 325 I=L1,L2	SPIN 70
CALL SCRAH(I,1)	6PIN 71
Z=-A(I)/HOLD(L)	6PIN 72
DO 324 J=L3,N2	SPIN 73
ZZ=Z+HOLD(J)	6PIN 74
A(J)=A(J)+ZZ	SPIN 75
IF(ORAB(A(J)).GT.EB+ABS(ZZ)) GO TO 324	6PIN 76
A(J)=ZERO	6PIN 77
324 CONTINUE	6PIN 78
325 CALL SCRAH(I,2)	6PIN 79
IF(N.GT.L2) GO TO 321	6PIN 80
GO TO 12	6PIN 81
C	6PIN 82
C DIVIDE BY DIAGONAL ELEMENTS.	6PIN 83
C	6PIN 84
46 N1=N+1	6PIN 85
DO 43 I=1,N	6PIN 86
CALL SCRAH(I,1)	6PIN 87
ZZ=A(I)	6PIN 88
DET = DET * ZZ	6PIN 89
DO 40 J =N1,N2	SPIN 90
40 A(J)=A(J)/ZZ	6PIN 91
43 CALL SCRAH(I,2)	6PIN 92
1000 RETURN	6PIN 93
END	6PIN 94

SUBROUTINE STATO	STAT 1
COMMON / BLOCKA/MODE,H,KARD(77),KARO,ARO,ARG2,NENCO(19),KROEND	STAT 2
1,NENCOS(19,5),KSAVE,NSAVE,HFLAG	STAT 3
COMMON / BLOCKD / RC(2499),IAROS(69),KIND(39),AROTAB(51),NMAX,	STAT 4
1,NRON,NCOL,NAROS,VWXYZ(5)	STAT 5
COMMON/BLOCKE/NH2(4),L1,L2,ISRFL0	STAT 6
DIMENSION AROS(1)	STAT 7
EQUIVALENCE(AROS(1),RC(2401))	STAT 8
DOUBLE PRECISION XX(3),X1,X2,X3	STAT 9
DIMENSION II(3),INC(3)	STAT 10
EQUIVALENCE(II,II(1)),(I2,II(2)),(I3,II(3)),(INC(1),IN1),	STAT 11
(INC(2),IN2),(INC(3),IN3),(XX(1),X1),(XX(2),X2),(XX(3),X3)	STAT 12
ZERO=0.	STAT 13
ONE=1.	STAT 14
IF(L2.LE.3.AND.NAROS.NE.2) GO TO 910	STAT 15
IF(L2.GE.13.AND.NAROS.NE.4) GO TO 910	STAT 16
IF(L2.GT.3.AND.L2.LT.13.AND.NAROS.NE.3) GO TO 910	STAT 17
CALL ADDRESSINAROS,IL)	STAT 18
IF(IL) 20,30,40	STAT 19
20 CALL ERROR(20)	STAT 20
RETURN	STAT 21
30 CALL ERROR(11)	STAT 22
RETURN	STAT 23
40 IL=IL+NMAX-1	STAT 24
NAROS=NAROS-1	STAT 25
DO 50 I=1,NAROS	STAT 26
INC(I)=1	STAT 27
CALL ADDRESS(1,II(1))	STAT 28
IF(II(1)) 45,30,50	STAT 29
45 II(1)=-II(1)	STAT 30
INC(1)=0	STAT 31
50 XX(I)=RC(II(1))	STAT 32
J=NAROS	STAT 33
CALL CHIND(J)	STAT 34
IF(NMAX.NE.0) GO TO 60	STAT 35
CALL ERROR(9)	STAT 36
RETURN	STAT 37
60 CALL FLAG	STAT 38
IF(HFLAG.EQ.1) RETURN	STAT 39
ASSIGN 100 TO INDEX1	STAT 40
IF(IJ.EQ.1) ASSIGN 100 TO INDEX1	STAT 41
GO TO (110,120,130,140,150,160,170,180,190,200,210,220,230,240,	STAT 42
1250,250,270,280),L2	STAT 43
80 I3=I3+IN3	STAT 44
X3=RC(I3)	STAT 45
05 I2=I2+IN2	STAT 46
X2=RC(I2)	STAT 47
20 I1=I1+IN1	STAT 48
X1=RC(I1)	STAT 49
GO TO INDEX1,(100,105)	STAT 50
100 CALL VECTOR(Y,IL)	STAT 51
RETURN	STAT 52
105 RC(IL)=Y	STAT 53
IL=IL+1	STAT 54

IF(IL.GT.JL2) RETURN	STAT 55
GO TO INDEX2.(111,121,131,141,151,161,171,181,191,201,211,221,231,	STAT 56
241,251,261,271,281)	STAT 57
903 CALL ERROR(103)	STAT 58
Y=ZERO	STAT 59
IF((L2-8)/5) 90,85,80	STAT 60
110 ASSIGN 111 TO INDEX2	STAT 61
111 Y=YORHX(X1)	STAT 62
GO TO 90	STAT 63
120 ASSIGN 121 TO INDEX2	STAT 64
121 IF(X1.LT.ZERO.OR.X1.GT.ONE) GO TO 903	STAT 65
Y=YORHX(X1)	STAT 66
GO TO 90	STAT 67
130 ASSIGN 131 TO INDEX2	STAT 68
131 Y=YORHX(X1)	STAT 69
GO TO 90	STAT 70
140 ASSIGN 141 TO INDEX2	STAT 71
141 IF(X1.LT.ZERO.OR.X2.LE.ZERO) GO TO 903	STAT 72
Y=ORHX(X1,X2)	STAT 73
GO TO 90	STAT 74
150 ASSIGN 151 TO INDEX2	STAT 75
151 IF(X1.LT.ZERO.OR.X1.GT.ONE.OR.X2.LE.ZERO) GO TO 903	STAT 76
Y=ORHX(X1,X2)	STAT 77
GO TO 90	STAT 78
160 ASSIGN 161 TO INDEX2	STAT 79
161 IF(X1.LT.ZERO.OR.X2.LE.ZERO) GO TO 903	STAT 80
Y=ORHX(X1,X2)	STAT 81
GO TO 90	STAT 82
170 ASSIGN 171 TO INDEX2	STAT 83
171 IF(X1.LT.ZERO.OR.X2.LE.ZERO) GO TO 903	STAT 84
Y=ORHX(X1,X2)	STAT 85
GO TO 90	STAT 86
180 ASSIGN 181 TO INDEX2	STAT 87
181 IF(X1.LT.ZERO.OR.X1.GT.ONE.OR.X2.LE.ZERO) GO TO 903	STAT 88
Y=ORHX(X1,X2)	STAT 89
GO TO 90	STAT 90
190 ASSIGN 191 TO INDEX2	STAT 91
191 IF(X1.LT.ZERO.OR.X2.LE.ZERO) GO TO 903	STAT 92
Y=ORHX(X1,X2)	STAT 93
GO TO 90	STAT 94
200 ASSIGN 201 TO INDEX2	STAT 95
201 IF(X2.LE.ZERO) GO TO 903	STAT 96
Y=TX(X1,X2)	STAT 97
GO TO 90	STAT 98
210 ASSIGN 211 TO INDEX2	STAT 99
211 IF(X1.LT.ZERO.OR.X1.GT.ONE.OR.X2.LE.ZERO) GO TO 903	STAT 100
Y=TX(X1,X2)	STAT 101
GO TO 90	STAT 102
220 ASSIGN 221 TO INDEX2	STAT 103
221 IF(X2.LE.ZERO) GO TO 903	STAT 104
Y=TX(X1,X2)	STAT 105
GO TO 90	STAT 106
230 ASSIGN 231 TO INDEX2	STAT 107
231 IF(X1.LT.ZERO.OR.X1.GT.ONE.OR.X2.LE.ZERO.OR.X3.LE.ZERO) GO TO 903	STAT 108

Y=BETAX(X1,X2,X3)	STAT 109
GO TO 80	STAT 110
240 ASSIGN 241 TO INDEX2	STAT 111
241 IF(X1.LT.ZERO.OR.X1.GT.ONE.OR.X2.LE.ZERO.OR.X3.LE.ZERO) GO TO 903	STAT 112
Y=BETAP(X1,X2,X3)	STAT 113
GO TO 80	STAT 114
250 ASSIGN 251 TO INDEX2	STAT 115
251 IF(X1.LT.ZERO.OR.X1.GT.ONE.OR.X2.LE.ZERO.OR.X3.LE.ZERO) GO TO 903	STAT 116
Y=BETA2(X1,X2,X3)	STAT 117
GO TO 80	STAT 118
260 ASSIGN 261 TO INDEX2	STAT 119
261 IF(X1.LT.ZERO.OR.X2.LE.ZERO.OR.X3.LE.ZERO) GO TO 903	STAT 120
Y=FFX(X1,X2,X3)	STAT 121
GO TO 80	STAT 122
270 ASSIGN 271 TO INDEX2	STAT 123
271 IF(X1.LT.ZERO.OR.X1.GT.ONE.OR.X2.LE.ZERO.OR.X3.LE.ZERO) GO TO 903	STAT 124
Y=FFP(X1,X2,X3)	STAT 125
GO TO 80	STAT 126
280 ASSIGN 281 TO INDEX2	STAT 127
281 IF(X1.LT.ZERO.OR.X2.LE.ZERO.OR.X3.LE.ZERO) GO TO 903	STAT 128
Y=FFZ(X1,X2,X3)	STAT 129
GO TO 80	STAT 130
810 CALL ERROR(10)	STAT 131
RETURN	STAT 132
END	STAT 133

SUBROUTINE TRANSF	TRAN 1
COMMON / BLOCKA/MODE,M,KARD(77),KARO,ARO,ARO2,NEICO(19),KROEND	TRAN 2
I,NEICOS(19,5),KSAVE,MSAVE,NFLAO	TRAN 3
COMMON / BLOCKO / RC(2439),IAROS(69),KIND(39),AROTAB(51),NRMAX,	TRAN 4
I,NROH,NCOL,NAROS,VWXYZ(5)	TRAN 5
COMMON/BLOCKE/NAME(4),L1,L2,ISKFLO	TRAN 6
COMMON/SCRAT/AL(80)	TRAN 7
DIENSION HOLO(90)	TRAN 8
C *****	TRAN 9
C THIS SUBROUTINE IS CALLED IN RESPONSE TO THE COMMANDS	TRAN 10
C M(X'AX) AND M(XAX').	TRAN 11
C *****	TRAN 12
C CHECK TO SEE IF WE HAVE CORRECT NUMBER OF ARGUMENTS	TRAN 13
C *****	TRAN 14
IF(NAROS.GT.10.OR.NAROS.LT.8) GO TO 210	TRAN 15
C *****	TRAN 16
C CHECK TO SEE IF ALL ARGUMENTS ARE INTEGERS	TRAN 17
C *****	TRAN 18
J=NAROS	TRAN 19
CALL CKIND(J)	TRAN 20
IF(J.NE.0) GO TO 030	TRAN 21
C *****	TRAN 22
C CHECK TO SEE IF DIMENSIONS ARE CORRECT	TRAN 23
C *****	TRAN 24
IF(NAROS.EQ.0) GO TO 230	TRAN 25
IF(NAROS.EQ.10) GO TO 200	TRAN 26
IF(IAROS(3).EQ.IAROS(8-L2)) GO TO 230	TRAN 27
GO TO 030	TRAN 28
200 IF(IAROS(3).EQ.IAROS(4).AND.IAROS(9).EQ.IAROS(8-L2)) GO TO 230	TRAN 29
GO TO 030	TRAN 30
C *****	TRAN 31
C CHECK TO SEE IF DIMENSIONS ARE OUT OF RANGE AND	TRAN 32
C FIND ADDRESSES OF MATRICES.	TRAN 33
C *****	TRAN 34
230 IF(NAROS-8) 240,230,320	TRAN 35
240 IAROS(10)=IAROS(8)	TRAN 36
IAROS(9)=IAROS(7)	TRAN 37
IAROS(6+L2)=IAROS(6)	TRAN 38
KIND(10)=0	TRAN 39
IF(L2.EQ.1) GO TO 250	TRAN 40
IAROS(7)=IAROS(9)	TRAN 41
GO TO 260	TRAN 42
250 IAROS(10)=IAROS(7)	TRAN 43
260 IAROS(6)=IAROS(5)	TRAN 44
IAROS(5)=IAROS(4)	TRAN 45
IAROS(4)=IAROS(3)	TRAN 46
GO TO 300	TRAN 47
280 DO 300 I=1,7	TRAN 48
III=10-I	TRAN 49
300 IAROS(III+1)=IAROS(III)	TRAN 50
320 IAROS(11)=IAROS(L2+8)	TRAN 51
IAROS(12)=IAROS(11)	TRAN 52
J=9	TRAN 53
CALL MTRCHK(J)	TRAN 54

IF(J-1) 450,830,350	TRAN 55
350 CALL ERROR(17)	TRAN 56
RETURN	TRAN 57
210 CALL ERROR(10)	TRAN 58
RETURN	TRAN 59
450 CALL PLOX	TRAN 60
IF(INFLAG.EQ.1) RETURN	TRAN 61
INCA=1	TRAN 62
IF(L2.EQ.1) GO TO 80	TRAN 63
IENDI=IAROS(0)	TRAN 64
INCX=1	TRAN 65
INCH=NRON	TRAN 66
GO TO 90	TRAN 67
80 INCH=1	TRAN 68
INCX=NRON	TRAN 69
IENDI=IAROS(7)	TRAN 70
90 IF(IENDI.LE.15) GO TO 95	TRAN 71
CALL ERROR(24)	TRAN 72
RETURN	TRAN 73
95 MARKX=IAROS(5)	TRAN 74
IAROS0=IAROS(9)	TRAN 75
DO 120 J=1,IENDI	TRAN 76
MARKAH=IAROS(1)	TRAN 77
DO 110 I=1,IAROS0	TRAN 78
MARKA=MARKAH	TRAN 79
MARKX=MARKXN	TRAN 80
A(I)=0.	TRAN 81
DO 100 K=1,IAROS0	TRAN 82
A(I)=A(I)+RC(MARKX)+RC(MARKA)	TRAN 83
MARKX=MARKX+INCX	TRAN 84
100 MARKA=MARKA+INCA	TRAN 85
110 MARKAH=MARKAH+NRON	TRAN 86
CALL SCRAM(J,2)	TRAN 87
120 MARKXN=MARKXN+INCH	TRAN 88
DO 160 K=1,IENDI	TRAN 89
MARKXN=IAROS(6)	TRAN 90
CALL SCRAM(K,1)	TRAN 91
DO 150 I=1,IAROS0	TRAN 92
150 HOLD(I)=A(I)	TRAN 93
DO 140 J=1,IENDI	TRAN 94
MARKX=MARKXN	TRAN 95
A(J)=0.	TRAN 96
DO 140 I=1,IAROS0	TRAN 97
A(J)=HOLD(I)+RC(MARKX)+A(I)	TRAN 98
140 MARKX=MARKX+INCX	TRAN 99
160 MARKXN=MARKXN+INCH	TRAN 100
180 CALL SCRAM(K,1)	TRAN 101
K=IENDI+1	TRAN 102
C *****	TRAN 103
C STORE RESULTS IN WORKSHEET	TRAN 104
C *****	TRAN 105
ICH=IAROS(9)	TRAN 106
DO 820 J=1,IENDI	TRAN 107
CALL SCRAM(K,1)	TRAN 108

```
IC=ICH  
DO 800 I=1,ICNDI  
RC(1C)=A(1)  
IC=IC+NRON  
800 CONTINUE  
ICH=ICH+1  
820 K=K+1  
GO TO 840  
830 CALL ERROR(3)  
840 RETURN  
END
```

```
TRAN 109  
TRAN 110  
TRAN 111  
TRAN 112  
TRAN 113  
TRAN 114  
TRAN 115  
TRAN 116  
TRAN 117  
TRAN 118  
TRAN 119
```

	SUBROUTINE VARCON(NAME)	VARC	1
	COMMON / BLOCKA/MODE,M,KARD(77),KARG,HRO,ARG2,NEHCD(10),KRDENO	VARC	2
	1,NEHCD5(10,5),KSAVE,NSAVE,NFLAO	VARC	3
	DIMENSION NAME(2),N(12)	VARC	4
	DATA N(1),N(2),N(3),N(4),N(5),N(6),N(7),N(8),N(9),N(10),N(11),	VARC	5
	1 N(12)/10705,10030,16767,17493,19223,18954,1977,5*0/	VARC	6
		VARC	7
	THIS SUBROUTINE IS USED TO LOCATE ONE OF THE VARIABLES:	VARC	8
	NRMAX,V,H,X,Y,Z	VARC	8
		VARC	10
	DO 10 IM=1,6	VARC	11
	I = IM	VARC	12
	IF(NAME(1).EQ.N(I).AND.NAME(2).EQ.N(I+6))GO TO 20	VARC	13
10	CONTINUE	VARC	14
	I=0	VARC	15
20	ARG=I	VARC	16
	RETURN	VARC	17
	END	VARC	18
	SUBROUTINE VECTOR(A, J)	VECT	1
	COMMON / BLOCKD / RC(2499),IARG5(69),KIND(39),ARCTAO(51),NRMAX,	VECT	2
	1 NROW,NCOL,NAROS,VWXYZ(6)	VECT	3
		VECT	4
	THIS SUBROUTINE STORES THE VALUE OF A IN	VECT	5
	EACH ROW OF THE COLUMN STARTING AT J	VECT	6
	DOWN TO NRMAX.	VECT	7
		VECT	8
	IF(NRMAX .EQ. 0) GO TO 20	VECT	9
	K = J + NRMAX - 1	VECT	10
	DO 10 I = J, K	VECT	11
10	RC(I) = A	VECT	12
20	RETURN	VECT	13
	END	VECT	14

	SUBROUTINE HORKD (N)	WORK	1
	COMMON / BLOCKD / RC(2499),IARCS(69),KIND(99),NRCTAB(51),NRMAX,	WORK	2
	1 NRON,NCOL,NARCS,VNXYZ(5)	WORK	3
	COMMON KEY,IOVLY,ITYPE	WORK	4
	DIMENSION IRAY(9),JRAY(5),SET(0.5),TEXT(100)	WORK	5
C		WORK	6
C	THIS SUBROUTINE IS USED TO DISPLAY A SECTION OF THE	WORK	7
C	WORKSHEET ON THE 2250 SCREEN.	WORK	8
C		WORK	9
	LT(I,J)=(J-1)*80+I	WORK	10
	NOUN=4	WORK	11
	1 KEY=KEY-3	WORK	12
	CALL DEGRAS(100)	WORK	13
	WRITE(NOUN,90) KEY	WORK	14
	90 FORMAT(/,20X,'WORKSHEET PART',19)	WORK	15
	CALL FETCH(TEXT,NCF,499)	WORK	16
	CALL GROPY(TEXT,NCF,499)	WORK	17
	IA=41	WORK	18
	IF(KEY.LT.10) IA=1	WORK	19
	ID=IA+39	WORK	20
	JA=5+MOD(KEY-1,6)+1	WORK	21
	JB=JA+4	WORK	22
	JY=0	WORK	23
	DO 9 JZ=JA,JB	WORK	24
	JY=JY+1	WORK	25
	9 JRAY(JY)=JZ	WORK	26
	WRITE(NOUN,91) JRAY	WORK	27
	91 FORMAT(/,30X,'C O L U M N S'/17,4114)	WORK	28
	CALL FETCH(TEXT,NCF,499)	WORK	29
	CALL GROPY(TEXT,NCF,499)	WORK	30
	DO 9 I=IA,IB,9	WORK	31
	IPT=I+7	WORK	32
	IY=0	WORK	33
	DO 10 IX=I,IPT	WORK	34
	IY = IY + 1	WORK	35
	IRAY(IY)=IX	WORK	36
	JY=0	WORK	37
	DO 10 J=JA,JB	WORK	38
	JY=JY+1	WORK	39
	10 GET(IY,JY)=RC(LT(IX,JY))	WORK	40
	WRITE(NOUN,92) (IRAY(IY),(GET(IH,JH),JH=1,5),IH=1,9)	WORK	41
	92 FORMAT(12,5G14.6)	WORK	42
	CALL FETCH(TEXT,NCF,499)	WORK	43
	CALL GROPY(TEXT,NCF,499)	WORK	44
	9 CONTINUE	WORK	45
	CALL GROPY(' ',1,499)	WORK	46
	CALL GROPY(' ',1,499)	WORK	47
	CALL GROPY(' ',1,499)	WORK	48
	99 RETURN	WORK	49
	END	WORK	50

	SUBROUTINE XECUTE	XECU	1
	COMMON / BLOCKN/NOOE,N,KARD(77),KARG,ARG,ARG2,NENCO(19),KRDENO	XECU	2
	1,NENCOG(19,5),KSAVE,NSAVE,NFLAO	XECU	3
	COMMON/BLOCKF/NONE(3),L1,L2,IGRFLD	XECU	4
	COMMON KEY,IOVLY,ITYPE	XECU	5
C		XECU	6
C	THIS SUBROUTINE IS USED TO PASS CONTROL TO THE SUBROUTINE TO BE	XECU	7
C	USED IN EXECUTING A PARTICULAR COMMAND.	XECU	8
C		XECU	9
	90 IF (L1 .LE. 90) GO TO (100,500,900,1100,1200,1300,1500,1600,1700,XECU	10	
	1,1800,2000,2100,2700,2300,3000),L1	XECU	11
	100 CALL RESET	XECU	12
	GO TO 9000	XECU	13
	500 CALL READX	XECU	14
	GO TO 9000	XECU	15
	900 CALL MXTX	XECU	16
	GO TO 9000	XECU	17
	1100 CALL ARITH	XECU	18
	GO TO 9000	XECU	19
	1200 CALL FUNCT	XECU	20
	GO TO 9000	XECU	21
	1500 GO TO (1901,1902),L2	XECU	22
	1901 CALL GENER	XECU	23
	GO TO 9000	XECU	24
	1902 CALL GET	XECU	25
	GO TO 9000	XECU	26
	1600 CALL HOP	XECU	27
	GO TO 9000	XECU	28
	1800 CALL INVERT	XECU	29
	GO TO 9000	XECU	30
	1700 IF(L2 .EQ. 2) GO TO 1720	XECU	31
	CALL HAULT	XECU	32
	GO TO 9000	XECU	33
	1720 CALL MRAICE	XECU	34
	GO TO 9000	XECU	35
	1900 CALL MATRIX	XECU	36
	GO TO 9000	XECU	37
	2000 CALL MSCRON	XECU	38
	GO TO 9000	XECU	39
	2100 GO TO (2101, 2101, 2103, 2104, 2104, 2104, 2104, 2100, 2100,	XECU	40
	1 2110, 2111, 2112, 2113,2100,9000),L2	XECU	41
	2101 CALL PRORCH	XECU	42
	GO TO 9000	XECU	43
	2103 CALL DEFINE	XECU	44
	GO TO 9000	XECU	45
	2104 CALL EXTREN	XECU	46
	GO TO 9000	XECU	47
	2109 CALL GORREN	XECU	48
	GO TO 9000	XECU	49
	2110 CALL ERASE	XECU	50
	GO TO 9000	XECU	51
	2111 CALL EXCHNO	XECU	52
	GO TO 9000	XECU	53
	2112 CALL FLIP	XECU	54

GO TO 9000	XECU 55
2119 CALL CHANGE	XECU 56
GO TO 9000	XECU 57
2300 GO TO 1 2310. 2310. 2310. 2310. 2310. 2320. 2320. 2320. 2320. 2330.	XECU 58
1 23001.L2	XECU 59
2310 CALL HISC2	XECU 60
GO TO 9000	XECU 61
2320 CALL MOVE	XECU 62
GO TO 9000	XECU 63
2330 CALL PROMOTE	XECU 64
GO TO 9000	XECU 65
2700 CALL EXPCON	XECU 66
GO TO 9000	XECU 67
9000 CALL STATO	XECU 68
9000 IF(INFLAG.EQ.1.OR.KEY.EQ.31) GO TO 9010	XECU 69
KSAVE=KSAVE+1	XECU 70
DO 9001 III=1.19	XECU 71
9001 NENCDS(III,KSAVE)=NENCDS(III)	XECU 72
IF(KSAVE.LT.5) GO TO 9010	XECU 73
IF(10VLY.GT.40) GO TO 9003	XECU 74
9004 WRITE(KSAVE*10VLY) NENCDS	XECU 75
KSAVE=0	XECU 76
9010 RETURN	XECU 77
9003 CALL PROGRAM(1)	XECU 78
IF(KEY.EQ.31) RETURN	XECU 79
10VLY = 1	XECU 80
GO TO 9004	XECU 81
END	XECU 82

```

SUBROUTINE XOHNIT
COMMON / BLOCKA/MODE,H,KARD(77),KARO,ARG,ARG2,NENCD(19),KROEND
1,NENCDS(19,5),KSAVE,NSAVE,NFLAO
COMMON / BLOCKB / RC(2459),IARG3(69),KIND(99),ARCTAD(51),NRMAX,
1 NRON,NCOL,NARG6,VNXYE(5)
COMMON KEY,IOVLY,ITYPE

```

C
C
C

THIS SUBROUTINE INITIALIZES SOME CONSTANTS WHICH WILL BE
USED BY THE SYSTEM.

```

IOVLY=1
KSAVE=0
MODE=1
NRMAX=0
RETURN
END

```

```

XOHN 1
XOHN 2
XOHN 3
XOHN 4
XOHN 6
XOHN 6
XOHN 7
XOHN 8
XOHN 8
XOHN 10
XOHN 11
XOHN 12
XOHN 13
XOHN 14
XOHN 15
XOHN 16

```

	SUBROUTINE XPND(T , K , Y , KND)	XPND 1
	COMMON / BLOCKD / RC(2499),IARG(69),KIND(99),AROT(51),NRMAX,	XPND 2
	1 ARDH,NCOL,NARCS,VWXYZ(5)	XPND 3
	DIMENSION T(2)	XPND 4
C		XPND 5
C	THIS SUBROUTINE IS USED TO GET Y TO THE VALUE INDICATED BY T.	XPND 6
C	KND IS USED TO INDICATE THE TYPE OF THE ARGUMENT(REAL=1,INTEGER=0)	XPND 7
C		XPND 8
C	K IS USED TO INDICATE EITHER AN ERROR OR THE LENGTH OF THE ARC	XPND 9
C		XPND 10
C	K IS RETURNED 0 IF ARC IN STATEMENT IS ONE WORD LONG	XPND 11
C	K IS RETURNED 1 IF ARC IN STATEMENT IS TWO WORDS LONG	XPND 12
C	K IS RETURNED -(ERROR NUMBER) IF ERROR OCCURS.	XPND 13
C		XPND 14
	IT = -T(1)	XPND 15
	IF(IT .LT. 10) GO TO 60	XPND 16
C		XPND 17
C	'ROW,COL' ENTRY	XPND 18
C		XPND 19
	IT = IT - 0200	XPND 20
	IF(IT .GT. 0 .AND. IT .LE. NROW) GO TO 41	XPND 21
	K = -10	XPND 22
	GO TO 44	XPND 23
	41 IARG(69) = ARG(T(2)) - 0102.	XPND 24
	KIND(69) = 0	XPND 25
	CALL ARGES(69 , J)	XPND 26
	IF(J .NE. 0) GO TO 40	XPND 27
	K = -11	XPND 28
	44 RETURN	XPND 29
	40 J = J + IT	XPND 30
	KND = 0	XPND 31
	IF(T(2) .LT. 0) KND = 1	XPND 32
	Y = RC(J - 1)	XPND 33
	K = 1	XPND 34
	GO TO 44	XPND 35
C		XPND 36
C	NRMAX , V , W , X , Y , Z . REFERENCE.	XPND 37
C		XPND 38
	60 IU = IT / 2	XPND 39
	KND = IT - 2 * IU	XPND 40
	K = 0	XPND 41
	IF(IU .LE. 1) GO TO 70	XPND 42
	Y = VWXYZ(IU-1)	XPND 43
	GO TO 44	XPND 44
	70 Y = NRMAX	XPND 45
	GO TO 44	XPND 46
	END	XPND 47

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) University of Georgia Department of Statistics and Computer Science		2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	
		2b. GROUP UNCLASSIFIED	
3. REPORT TITLE AN INTERACTIVE WORKSHEET SYSTEM FOR STATISTICAL USAGE			
4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)			
5. AUTHOR(S) (First name, middle initial, last name) Stephen F. Bingham and Rolf E. Bargmann			
6. REPORT DATE August 1975		7a. TOTAL NO. OF PAGES 291	7b. NO. OF REFS 63
8a. CONTRACT OR GRANT NO. N00014-69-A-0423		9a. ORIGINATOR'S REPORT NUMBER(S) Themis Report #31	
8b. PROJECT NO. NR024-261		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) Department of Statistics Rept. #106	
10. DISTRIBUTION STATEMENT Reproduction in whole or in part is permitted for any purpose of the United States Government, Office of Naval Research, Washington, D.C.			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY	
13. ABSTRACT This report discusses the implementation of an interactive version of the National Bureau of Standard's OMNITAB system. This version has been adopted to work under a Graphics Monitor System on an IBM 2250 terminal, connected to an IBM 360 or 370 central processor. Several routines have been added or adapted which make the system especially useful for statistical applications, and as an instructional tool. The immediate availability of displays of sections of the worksheet, after each instruction, is the central feature of this adaptation. Several examples of statistical applications are included in this report.			

UNCLASSIFIED

Security Classification

A-31408

UNCLASSIFIED

Security Classification

14.

KEY WORDS

LINK A

LINK B

LINK C

ROLE

WT

ROLE

WT

ROLE

WT

Computer Graphics
Conversational Computation
Interactive Programming
Multivariate Analysis
OMNITAB
Statistics
Worksheet System

DD FORM 1473 (BACK)
1 NOV 63

101-487-0521

UNCLASSIFIED

Security Classification

101-487-0521